RECURRENT NEURAL NETWORKS FOR REPRESENTING, SEGMENTING, AND CLASSIFYING SURGICAL ACTIVITIES

by

Robert DiPietro

A dissertation submitted to Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy

Baltimore, Maryland

January, 2020

Abstract

Robot-assisted surgery has enabled scalable, transparent capture of high-quality data during operation, and this has in turn led to many new research opportunities. Among these opportunities are those that aim to improve the objectivity and efficiency of surgical training, which include making performance assessment and feedback more objective and consistent; providing more specific or *localized* assessment and feedback; delegating this responsibility to machines, which have the potential to provide feedback in any desired abundance; and having machines go even further, for example by optimizing practice routines, in the form of a virtual coach. In this thesis, we focus on a foundation that serves all of these objectives: automated surgical activity recognition, or in other words the ability to automatically determine *what* activities a surgeon is performing and *when* those activities are taking place.

First, we introduce the use of recurrent neural networks (RNNs) for localizing and classifying surgical activities from motion data. Here, we show for the first time that this task is possible at the level of maneuvers, which unlike the activities considered in prior work are already a part of surgical training curricula. Second, we study the

ABSTRACT

ability of RNNs to learn dependencies over extremely long time periods, which we posit are present in surgical motion data; and we introduce MIST RNNs, a new RNN architecture that is capable of capturing these extremely long-term dependencies. Third, we investigate unsupervised learning using surgical motion data: we show that predicting future motion from past motion with RNNs, using motion data alone, leads to meaningful and useful representations of surgical motion. This approach leads to the discovery of surgical activities from unannotated data, and to state-of-the-art performance for querying a database of surgical activity using motion-based queries. Finally, we depart from a common yet limiting assumption in nearly all prior work on surgical activity recognition: that annotated training data, which is difficult and expensive to acquire, is available in abundance. We demonstrate for the first time that both gesture recognition and maneuver recognition are feasible even when very few annotated sequences are available; and that future-prediction based representation learning, prior to the recognition phase, yields significant performance improvements when annotated data is scarce.

Primary Advisor: Gregory D. Hager (Johns Hopkins University)

Co-Advisor: Nassir Navab (Johns Hopkins University)

Reader: Austin Reiter (Facebook AI Research)

Acknowledgments

I am extremely grateful to everyone who helped me through the PhD – in terms of research, in terms of teaching, and in terms of life itself, outside of the PhD program.

First, I'd like to thank my advisor, Gregory D. Hager, for his continued support and guidance throughout my entire PhD. Both inside and outside of research, Greg is extremely supportive of any ideas that excite his students. I couldn't be more grateful for this. During my PhD, he let me spend over a year in Munich; pursue multiple internships; and teach and TA many different courses at Hopkins. It is hard to portray just how important these 'extra' experiences were, and I can't thank Greg enough for not only allowing but *encouraging* these opportunities. In addition, Greg has an amazing ability to understand what deserves focus and what does not; and meeting with Greg has helped push my research forward time and time again.

I'd also like to thank the other members of my committee, Nassir Navab and Austin Reiter. Nassir and Austin have both served as great mentors throughout long stretches of my PhD, and I am honored to have them both on my committee. Some of the most memorable moments during my PhD occurred in Munich, while visiting

Nassir's lab at TU München.

I'm also grateful to everyone that I've had the chance to collaborate with during my time at Hopkins. I'd especially like to thank Narges Ahmidi, Anand Malpani, and Swaroop Vedula. Narges initiated my excitement for surgical activity recognition; and Narges, Anand, and Swaroop have all played a major role in my understanding of our work from a clinical perspective. I'd also like to thank my other collaborators, and in particular my other co-authors, including Colin Lea, Madeleine Waldram, Gyusung Lee, Mija Lee, Christian Rupprecht, Iro Laina, Huseyin Coskun, Felix Achilles, Federico Tombari, Maximilian Baust, Ralf Stauder, Ergün Kayis, Armin Schneider, Michael Kranzfelder, and Hubertus Feussner.

In addition to research, teaching was a memorable and important part of my PhD. I especially thank Joanne Selinski for this, who gave me the chance to teach on many occasions. Joanne is an excellent teacher, and I enjoyed learning from her. In addition, I'd like to thank Yair Amir. Yair has an incredibly unique and successful teaching style, and learning from him while co-teaching his course on intermediate programming was extremely valuable.

I'd also like to thank all of the staff who helped me so much over the years – especially Deborah DeFord, Cathy Thornton, Tracy Marshall, and Laura Graham at Hopkins; and Martina Hilla at TU München.

Next I'd like to thank all of those who helped me outside of the PhD program itself. I thank my girlfriend, You Ri Hwang, for being so positive and supportive, and

for continuously reminding me of life outside of the PhD. I thank all of my friends for our memorable experiences, from getting a coffee or taking a walk to participating in the Christkindlmarktchallenge. And I thank my mother and father, Patricia E. DiPietro and Robert J. DiPietro, and my grandfather, Carmine DiPietro, for being supportive throughout my entire life, and also for giving me the chance to tinker with computers at a very young age.

Finally, I'd like to thank the numerous agencies that provided funding throughout my PhD. This funding included a fellowship from Intuitive Surgical; grant 291763 from the Technische Universität München – Institute for Advanced Study, German Excellence Initiative and the European Union Seventh Framework Programme; grant R01-DE025265 from the National Institutes of Health; grant OISE-1065092 from the National Science Foundation; grant IIS-1900952 from the National Science Foundation; and a fellowship for Modeling, Simulation, and Training from the Link Foundation.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this introductory chapter we provide motivation for this thesis and a general outline of the material that follows. First, we motivate surgical activity recognition from a clinical perspective. Next, we highlight the primary questions that are asked and addressed by this thesis; we point to the chapters that are relevant to each question; and we formulate a concise thesis statement that summarizes our major findings. Finally, we list the primary contributions of the thesis, alongside the publications that have resulted from these contributions.

1.1 Motivation

Superior technical skill in the operating room is associated with better patient outcomes [1, 2], and at the core of surgical education is the belief that technical skill

is improved through deliberate practice and appropriate feedback [3, 4]. In the realm of surgical training, however, feedback comes from expert surgeons, whose time is understandably limited; and current standards for providing technical skills training are constrained by time [5]. In addition, we should strive for both performance assessment and feedback to be objective, repeatable, and localized, in the sense of being specific rather than general. However, in current practice, most available methods for surgical skill assessment are subjective and global [6]. In the context of deliberate practice, this means that trainees are often left guessing at which exercises to focus on; or, even when this is clear, left guessing at which aspects of their handling are responsible for suboptimal performance. This is especially true in practice sessions that take place outside of the operating room, which are an important part of current training curricula [7, 8], for example in a benchtop setting. Here, trainees perform repeated exercises involving activities that are common across surgical tasks, such as suturing and knot tying [7]; but during these sessions, an expert surgeon is often not available to provide feedback.

Current research objectives therefore aim to deliver automated, objective, and localized (or *targeted*) feedback with machines [9], [10], [11], [4], [12]. These research directions are especially promising within the realm of robot-assisted surgery: unlike traditional surgery, robot-assisted surgery enables the collection of high-quality surgical data to be collected over time in a transparent and automated fashion. This data can then be analyzed in detail, potentially even in real time.

In this thesis, we focus primarily on *automated surgical activity recognition*, which is the task of jointly *localizing* and *classifying* surgical activities over time. Surgical activity recognition serves as a foundation for all of the above objectives: if we are able to determine *what* activities are being performed, and *when* those activities are being performed, then each activity can be analyzed in isolation, thus enabling activity-specific analysis, performance assessment, and feedback. In addition, analyzing activities in isolation is simpler than analyzing longer sequences of activities over time, and so the development of *objective* assessment and feedback becomes more feasible. Finally, even *global* assessment and feedback can be enhanced, as the joint localization and segmentation of subactivities which compose global activity can be analyzed in terms of the high-level structure formed by these subactivities.

We also remark that automated surgical activity recognition is already the subject of much prior work [13–21]. As we progress through the thesis, we will contrast the differences with prior work in detail.

1.2 Outline

In this section we describe the main questions that aim to address in this thesis; and for each question we highlight the chapters that are most relevant.

1.2.1 Is it possible to recognize surgical activities at the level of maneuvers which, unlike the granularities considered in prior work, are already familiar to surgeons?

Maneuvers, which include high-level activities such as *knot tying* and *suture throw*, are already part of surgical training curricula [7]. However, prior work has considered only lower-level *gestures*, which exhibit very small time scales and less overall complexity, but which are foreign to surgeons. Leveraging recurrent neural networks for maneuver recognition is the subject of Chapter 3.

1.2.2 How critical is the modeling of *structure between activities* for surgical activity recognition?

From a technical rather than clinical perspective, we remark that nearly all prior work in surgical activity recognition is based on hidden-Markov-model (HMM) and conditional-random-field (CRF) based methods, with an emphasis on modeling the probabilistic structure that govern interactions between activities. Is this structure important for recognition, or should we shift our attention elsewhere? This question

is also addressed in Chapter 3.

1.2.3 How critical is the modeling of long-term dynamics for surgical activity recognition?

Maneuvers occur over 10s of seconds, and this thesis focuses on recurrent-neural-network (RNN) based methods for activity recognition, which are known to be limited with respect to their ability to learn long-term dependencies. Is it possible to create an RNN architecture that is capable of learning extremely long-term dependencies? And if so, does this capability improve surgical-activity-recognition performance? These questions are addressed in Chapter 4.

1.2.4 Is it possible to learn meaningful representations of surgical motion using *only motion data itself*, without any manual annotations?

Manual annotations are difficult and expensive to acquire, especially at scale. Is it possible to make use of surgical motion data even when no annotations are available? For example, is it possible to *discover* high-level surgical activities in a completely unsupervised fashion, and it is possible to learn representations that enable trainees to query a database of motion using motion-based queries? This questions are addressed

in Chapter 5.

1.2.5 Is surgical activity recognition possible even when manual annotations are scarce?

Annotations are not only difficult to acquire, but also subjective, in that preference of definitions and granularities can vary among experts. Meanwhile, collecting annotations at any one definition and granularity requires a tremendous effort, and this definition and granularity cannot easily be changed once acquired. These issues are alleviated if surgical activity recognition is feasible with very few annotated sequences. Is this possible? This question is also addressed in Chapter 5.

1.3 Thesis Statement

Surgical motion data can be modeled with recurrent neural networks in order to localize and classify surgical activities in an automated fashion, even when annotated data is scarce.

1.4 Contributions

Chapter 2 provides a dissemination of recurrent neural networks for both discriminative and generative modeling. This material will be published as the following

book chapter:

- R. DiPietro and G.D. Hager, "Deep Learning: RNNs and LSTM," in Handbook of Medical Image Computing and Computer Assisted Intervention. Elsevier, 2020.

Chapter 3 applies recurrent neural networks to the task of surgical activity recognition from motion data, under the assumption that an abundance of annotated data is available for training, and shows for the first time that the recognition of high-level manuevers (e.g., *suture throw*) is possible. Preliminary experiments were published as a conference paper and later expanded and published as a journal paper:

- R. DiPietro, C. Lea, A. Malpani, N. Ahmidi, S. S. Vedula, G. I. Lee, M. R. Lee, and G.D. Hager, "Recognizing Surgical Activities with Recurrent Neural Networks," International Conference on Medical Image Computing and Computer-Assisted Intervention, 2016.

- R. DiPietro, N. Ahmidi, A. Malpani, M. Waldram, G. I. Lee, M. R. Lee, S. S. Vedula, and G.D. Hager, "Segmenting and Classifying Activities in Robot-Assisted Surgery with Recurrent Neural Networks," International Journal of Computer Assisted Radiology and Surgery, 2019.

Chapter 4 introduces a novel recurrent-neural-network architecture that is capable of learning dependencies across extremely long time scales. This material was presented as a workshop paper:

- R. DiPietro, C. Rupprecht, N. Navab, and G.D. Hager, "Analyzing and Exploiting NARX Recurrent Neural Networks for Long-Term Dependencies," International Conference on Learning Representations Workshop, 2017.

Chapter 5 shows that it is possible to learn meaningful representations from surgical motion data alone, without any manual annotations, and that these representations can be leveraged for the downstream task of data-efficient surgical activity recognition. This material was published as two conference papers and serves as the basis for a journal paper that is in preparation:

- R. DiPietro and G.D. Hager, "Unsupervised Learning for Surgical Motion by Learning to Predict the Future," in International Conference on Medical Image Computing and Computer-Assisted Intervention, 2018.

- R. DiPietro and G.D. Hager, "Automated surgical activity recognition with one labeled sequence," in International Conference on Medical Image Computing and Computer-Assisted Intervention, 2019.

- R. DiPietro and G.D. Hager, "Data-Efficient Segmentation and Classification of Time-Series Data with Deep Future Prediction," in preparation, targeting Pattern Analysis and Machine Intelligence, 2020.

Other collaborative efforts that have helped shape the ideas in this thesis include

- R. DiPietro, R. Stauder, E. Kayis, A. Schneider, M. Kranzfelder, H. Feussner, G.D. Hager, and N. Navab, "Automated Surgical-Phase Recognition Using

Rapidly-Deployable Sensors," in the M2CAI Workshop at MICCAI, 2015.

- C. Rupprecht, I. Laina, R. DiPietro, M. Baust, F. Tombari, G.D. Hager, and N. Navab, "Learning in an Uncertain World: Representing Ambiguity Through Multiple Hypotheses," in IEEE International Conference on Computer Vision, 2017.

- H. Coskun, F. Achilles, R. DiPietro, N. Navab, and F. Tombari, "Long Short-Term Memory Kalman Filters: Recurrent Neural Estimators for Pose Regularization," in IEEE International Conference on Computer Vision, 2017.

Chapter 2

Background

In this chapter, we provide an overview of the primary technical tool of this thesis: recurrent neural networks (RNNs). First, we formulate RNNs as a simple extension to feedforward neural networks, introduced in order to handle sequential data. Next, we discuss the representation power of RNNs, but also discuss the distinct difference between representation and the ability to *learn* these representations from data. Finally, we discuss how RNNs can be used both for discriminative modeling, in which the aim to directly model a conditional distribution, and for generative modeling, in which the aim is to capture a distribution of observations themselves. Discriminative modeling and generative modeling are both used extensively in the thesis.

2.1 Recurrent Neural Networks

Recurrent neural networks (RNNs) are a class of neural networks that are naturally suited to processing time-series data and other sequential data. Here we introduce recurrent neural networks as an extension to feedforward networks, in order to allow the processing of variable-length (or even infinite-length) sequences, and some of the most popular recurrent architectures in use, including long short-term memory (LSTM) and gated recurrent units (GRUs).

2.1.1 From Feedforward to Recurrent

Recurrent neural networks are perhaps most easily understood by transitioning from typical feedforward networks. Feedforward networks traditionally map from fixed-size inputs to fixed-size outputs, for example to map from an image of fixed spatial extent to its class, or to a segmentation map of the same spatial extent. In contrast, recurrent neural networks naturally operate on variable-length input sequences and map to variable-length output sequences, for example to map from an image to various sentences that describe that image. This capability is achieved by sharing parameters and transformations over time. See Figure 2.1.

We proceed with an example. Consider a scenario in which we wish to map from a set of measurements describing a tumor, $\mathbf{x}$, to probability of malignancy, $p(y \mid \mathbf{x})$. Suppose we wish to use feedforward neural networks to solve this problem, and that

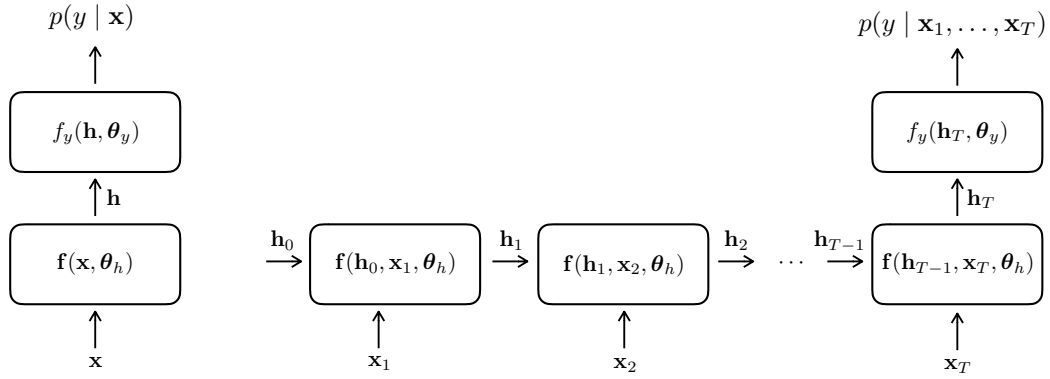


Figure 2.1: **An example feedforward network (left) and an example recurrent neural network (right).** In the recurrent example, the function $\mathbf{f}$ and its parameters $\boldsymbol{\theta}_h$ are shared over time.

we begin with the simple network

$$\mathbf{h} = \tanh(\mathbf{W}_{xh}\mathbf{x} + \mathbf{b}_h) \tag{2.1}$$

$$p(y \mid \mathbf{x}) = \sigma(\mathbf{W}_{hy}\mathbf{h} + b_y) \tag{2.2}$$

where all weights $\mathbf{W}$ and all biases $\mathbf{b}$ are parameters to be learned, tanh is the hyperbolic tangent function, and σ is the sigmoid function, the output of which is between 0 and 1. Given this network and a training set of ground-truth $(\mathbf{x}_i, y_i)$ pairs, say one per patient, we typically proceed by maximizing the conditional log likelihood $\sum_i \log p(y_i \mid \mathbf{x}_i)$. (This is equivalent to minimizing the overall cross entropy between our predicted distributions, given by the network, and the ground-truth distributions, given by our training data, as usually done in practice.) Optimization is typically carried out using a variant of stochastic gradient descent, and this is usually carried out in practice by 1. forming a computation graph that corresponds to this network

(and loss function) and 2. computing gradients via backpropagation [22].

We now ask the following question: what if each patient is not associated with a single measurement vector, but rather with a *sequence* of measurement vectors, each associated with a single examination? How can we modify the network above to map from a sequence of measurement vectors to a single probability of malignancy, corresponding to the latest examination date?

Perhaps the most straightforward approach is to stack each patient's measurement vectors into one, forming a new input of increased dimensionality. However, this approach is unsatisfactory for a number of reasons. First, we note that such a model makes no attempt to capture the inductive biases we might hope for in our model: it would treat all elements across all time steps in exactly the same way, rather than incorporating any explicit mechanism to capture local dynamics[1] Second, we expect the number of examinations per patient to vary, and as a reminder, the input to our network, $\mathbf{x}$, has fixed dimensionality. Thus if we proceed in this way, we will be forced to proceed with heuristics, either by throwing away past information for some patients (those with many examinations) or by padding the inputs of other patients (those with fewer examinations).

Instead, let's proceed by modifying the simple network above to carry over latent information from time step to time step (after training, these latent states can be

[1]It is interesting to note that this implicit bias is absent from modern transformer networks [23], which must learn any temporal correlations from scratch. Transformers have been remarkably successful in natural language processing, but they are typically trained with datasets that are orders of magnitude larger than any of the datasets considered in this thesis.

interpreted as learned representations that are specific to our task). First, let's modify our notation so that the single-examination case more explicitly corresponds to a sequence of length 1:

$$\mathbf{h}_1 = \tanh(\mathbf{W}_{xh}\mathbf{x}_1 + \mathbf{b}_h) \tag{2.3}$$

$$p(y_1 \mid \mathbf{x}_1) = \sigma(\mathbf{W}_{hy}\mathbf{h}_1 + b_y) \tag{2.4}$$

Next, let's modify the linear transformation used in the hidden-state computation to depend not only our input $\mathbf{x}$ but also on information from the past, which is carried through the hidden state. We will set our previous hidden state to $\mathbf{0}$, which can be interpreted as carrying over no information from the past:

$$\mathbf{h}_0 = \mathbf{0} \tag{2.5}$$

$$\mathbf{h}_1 = \tanh(\mathbf{W}_{hh}\mathbf{h}_0 + \mathbf{W}_{xh}\mathbf{x}_1 + \mathbf{b}_h) \tag{2.6}$$

$$p(y_1 \mid \mathbf{x}_1) = \sigma(\mathbf{W}_{hy}\mathbf{h}_1 + b_y) \tag{2.7}$$

Notice that, from a modeling perspective, this network is precisely equivalent to our original network; it differs only in notation and operation counts (introduced by the unnecessary matrix-vector multiply involving $\mathbf{h}_0 = \mathbf{0}$). However, our network now

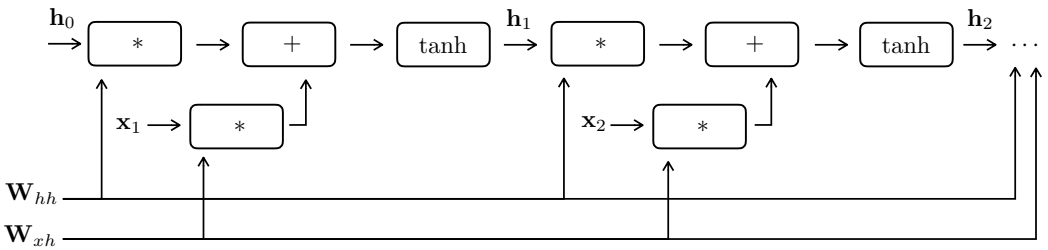


Figure 2.2: **A computation graph corresponding to a simple RNN.** The bias ($\mathbf{b}_h$) is omitted for simplicity. Note that all parameters, here $\mathbf{W}_{hh}$ and $\mathbf{W}_{xh}$, are shared over time. Two time steps are shown, but the computation graph can be unrolled indefinitely. The symbol $*$ denotes matrix multiplication (with implicit ordering assumed, e.g. $\mathbf{W}_{hh}\mathbf{h}_0$, not $\mathbf{h}_0\mathbf{W}_{hh}$).

naturally extends to patients with any number of examinations:

$$\mathbf{h}_0 = \mathbf{0} \tag{2.8}$$

$$\mathbf{h}_1 = \tanh(\mathbf{W}_{hh}\mathbf{h}_0 + \mathbf{W}_{xh}\mathbf{x}_1 + \mathbf{b}_h) \tag{2.9}$$

$$\mathbf{h}_2 = \tanh(\mathbf{W}_{hh}\mathbf{h}_1 + \mathbf{W}_{xh}\mathbf{x}_2 + \mathbf{b}_h) \tag{2.10}$$

$$\cdots \tag{2.11}$$

$$\mathbf{h}_T = \tanh(\mathbf{W}_{hh}\mathbf{h}_{T-1} + \mathbf{W}_{xh}\mathbf{x}_T + \mathbf{b}_h) \tag{2.12}$$

$$p(y_T \mid \mathbf{x}_1, \ldots, \mathbf{x}_T) = \sigma(\mathbf{W}_{hy}\mathbf{h}_T + b_y) \tag{2.13}$$

This RNN is capable of processing sequences of any length, for example one with a length of $T = 1$ and another with a length of $T = 7$, because *the transition function and its parameters are shared over time.*

Once defined, training the network is carried out in a way that is nearly identical to the process described above for feedforward networks. Each patient now corresponds

to a *sequence* of exams, $\mathbf{x}_1, \ldots, \mathbf{x}_T$, along with a label y_T. We form our computation graph by unrolling the RNN over these time steps and adding operatings to compute the loss; we obtain gradients via backpropagation; and we optimize via stochastic gradient descent.

Equations 2.9 – 2.12 represent of one of the first RNN variants that can be found in literature: that of *simple* RNNs or *Elman* RNNs [24]. In literature, we typically see the more compact representation

$$\mathbf{h}_t = \tanh(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{xh}\mathbf{x}_t + \mathbf{b}_h) \tag{2.14}$$

in which the initial hidden state is omitted, and unless otherwise specified is often assumed to be $\mathbf{h}_0 = \mathbf{0}$. A simplified computation graph corresponding to Equation 2.14 is shown in Figure 2.2.

2.1.2 On the Representation Power of RNNs

A well known result is that a feedforward network with only one hidden layer, under fairly unrestrictive assumptions and given enough hidden units, can approximate continuous functions to arbitrary accuracy [25]. The analagous result for simple RNNs is that they can approximate sequence-to-sequence mappings to arbitrary accuracy [26, 27]. (In fact, again under fairly unrestrictive assumptions, one can even show that simple RNNs are capable of simulating a Turing machine with only a small,

fixed number of units (886), under which binary streams are provided as the inputs and outputs of the RNN [28, 29].) Thus even this extremely simple architecture is very powerful from the perspective of *representation*.

However, in practice representational power is not our only concern, and it is important to note that these powerful results on representation in no way indicate that we can *learn* such representations from data in any reasonable amount of time. Indeed, we will later see that a major drawback of simple RNNs is their inability to learn long-term dependencies from data, and that long short-term memory (LSTM) was introduced explicitly to alleviate this issue.

2.1.3 More General RNNs

Simple RNNs map from hidden state to hidden state through Equation 2.14. More generally, RNNs typically map a sequence of inputs $\mathbf{x}_1, \ldots, \mathbf{x}_T$ to a sequence of hidden states $\mathbf{h}_1, \ldots, \mathbf{h}_T$ through a set of parameters $\boldsymbol{\theta}$ via

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t, \boldsymbol{\theta}) \tag{2.15}$$

where f is general, and in particular should *not* be interpreted as a simple activation function, e.g. tanh or σ. For example, in the next section, we will discuss long short-term memory, for which f is a composition involving various gates and nonlinearities.

In addition, the inputs $\mathbf{x}_t$ are not restricted to be our observed (input) data, and

similarly, $\mathbf{h}_t$ does not need to be related to our observed (output) data through a simple transformation, as seen in the motivating example above. From this perspective, RNNs are another building block that we can use to build rich models that can be trained in an end-to-end fashion. For example, we can compose what is often referred to as a *deep* RNN by applying Equation 2.15 multiple times: first, an RNN is applied to our observed inputs, yielding a sequence of hidden states; and these hidden states are fed as inputs into another RNN; and so on.

2.1.4 Long Short-Term Memory and Gated Recurrent Units

As mentioned earlier, although simple RNNs are powerful from the perspective of representation, we have no guarantees that we can *learn* such representations from data. Indeed, simple RNNs are known to be particularly susceptible to the so called *vanishing gradient problem* [30, 31]: gradient magnitudes relating a loss at time t to an input or hidden state at time $t-\tau$ fall of exponentially fast with τ, in turn making it extremely difficult to learn long-term dependencies.

Long short-term memory (LSTM) [32] was introduced to alleviate this issue, and has become one of the most popular RNN architectures to date. Gated recurrent units (GRUs) [33] were later introduced as a simpler alternative to LSTM, and have also become quite popular. Here we will introduce both architectures. The vanishing

gradient problem will be discussed at length in Chapter 4, where it will be discussed in the context of our work on Mixed History Recurrent Neural Networks.

We first remark that many different variants of LSTM and GRUs exist in literature, and that even the default implementations in various major deep learning frameworks often differ. (Performance is often similar – see, e.g., [34] – but this can nevertheless lead to confusion when reproducing results.) The most common variant of LSTM is described by

$$\mathbf{f}_t = \sigma(\mathbf{W}_{fh}\mathbf{h}_{t-1} + \mathbf{W}_{fx}\mathbf{x}_t + \mathbf{b}_f) \tag{2.16}$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_{ih}\mathbf{h}_{t-1} + \mathbf{W}_{ix}\mathbf{x}_t + \mathbf{b}_i) \tag{2.17}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{oh}\mathbf{h}_{t-1} + \mathbf{W}_{ox}\mathbf{x}_t + \mathbf{b}_o) \tag{2.18}$$

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_{ch}\mathbf{h}_{t-1} + \mathbf{W}_{cx}\mathbf{x}_t + \mathbf{b}_c) \tag{2.19}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \tag{2.20}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{2.21}$$

At a high level, LSTM is often interpreted as maintaining a memory cell, $\mathbf{c}_t$, which is reset, written to, and read from according to the forget gate $\mathbf{f}_t$, the input gate $\mathbf{i}_t$, and the output gate $\mathbf{o}_t$ [27], which all vary over time. We can see the clear similarity between the updates for these different gates; they differ only in that they have distinct weight matrices and biases (to be learned). In particular, all three gates very much resemble simple RNNs, but with the key difference being the sigmoid activation

function (restricting outputs to lie between 0 and 1), rather than the tanh activation function. These gates can therefore have elements that can be interpreted as lying between fully off (0), shutting down information flow; and fully on (1), allowing full information flow. Next, we see the formation of the *candidate* update to our memory cell, $\tilde{\mathbf{c}}_t$. Notice that this transformation is identical to that of a simple RNN. Next, in Equation 2.20, the elements of the previous cell $\mathbf{c}_{t-1}$ are combined with the candidate update $\tilde{\mathbf{c}}_t$, according according to the forget gate $\mathbf{f}_t$ and the input gate $\mathbf{i}_t$. And finally the new 'hidden state' $\mathbf{h}_t$ is formed by applying a final activation function to the memory cell and then weighting elements according to the output gate.

Equation 2.20, the formation of the new memory cell, is at the core of alleviating the vanishing gradient problem. Notice here that there is one path between $\mathbf{c}_{t-1}$ and $\mathbf{c}_t$ that is modulated *only* by the forget gate. We will explore this issue in detail in Chapter 4; for now we highlight the main idea. We have exponentially many paths from $\mathbf{c}_{T-\tau}$ to $\mathbf{c}_T$, but one of these paths corresponds simply to element-wise multiplications by forget gates. This in turn means that we have an additive gradient component which itself is the product of diagonal Jacobians with diagonal elements corresponding to the forget gates. Hence gradient contributions can still decay exponentially with τ, but *if the forget gates have elements that are close to 1, then the base of the exponential decay is also close to 1.* This is precisely why it is common in practice to initialize the bias $\mathbf{b}_f$ of the forget gate to be positive (e.g., 1 or 2) at the start of training.

Before moving on to GRUs, we make a few closing remarks about LSTM in hope of avoiding confusion. First, our notation was chosen to be consistent with literature. However, note that the true hidden state – the collection of all information passed from one time step to the next – is actually a concatenation of $\mathbf{h}_t$ and $\mathbf{c}_t$. Second, it is worth noting that some sources state that LSTM 'solves' the vanishing gradient problem and/or that LSTM alleviates the exploding-gradient problem. Both of these statements are incorrect. Gradient norms can still decay exponentially fast with delay, and with regard to the second statement, we know of no theoretical argument or empirical result that suggests that LSTM alleviates the exploding-gradient problem, and indeed many practitioners still find it necessary to clip gradients when training LSTM networks.

After encountering LSTM, one may wonder what parts of LSTM are most necessary, and whether it is possible to simplify the architecture while retaining its key benefits. One recently proposed architecture that was inspired by LSTM is that of gated recurrent units (GRUs) [33]. GRUs are simpler than LSTM, as they use one less gate and eliminate the need for distinguishing between hidden states and memory

cells. The most common variant is described by

$$\mathbf{u}_t = \sigma(\mathbf{W}_{uh}\mathbf{h}_{t-1} + \mathbf{W}_{ux}\mathbf{x}_t + \mathbf{b}_u) \tag{2.22}$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_{rh}\mathbf{h}_{t-1} + \mathbf{W}_{rx}\mathbf{x}_t + \mathbf{b}_r) \tag{2.23}$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_{hh}(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{W}_{hx}\mathbf{x}_t + \mathbf{b}_h) \tag{2.24}$$

$$\mathbf{h}_t = \mathbf{u}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{u}_t) \odot \tilde{\mathbf{h}}_t \tag{2.25}$$

Notice that GRUs mitigate the vanishing gradient problem using a mechanism very similar to that of LSTM; from Equation 2.25, we can see that if $u_{t,i}$ is nearly 1, then the corresponding partial $\frac{\partial h_{t,i}}{\partial h_{t-1,i}}$ is also nearly 1. $\mathbf{r}_t$ and $\mathbf{u}_t$ are referred to as the reset and update gates.

2.2 Modeling with RNNs

RNNs were motivated above through a simple example: mapping from a sequence of measurement vectors describing a tumor, $\mathbf{x}_1$ through $\mathbf{x}_T$, to the probability of malignancy as of the last measurement, $p(y_T \mid \mathbf{x}_1, \ldots, \mathbf{x}_T)$. This was an example of a *discriminative* model, and in particular a discriminative model which maps from *a sequence of inputs* to *a single output*. However, RNNs are also applicable to many other modeling scenarios, and this section aims to introduce the scenarios that we encounter throughout the thesis.

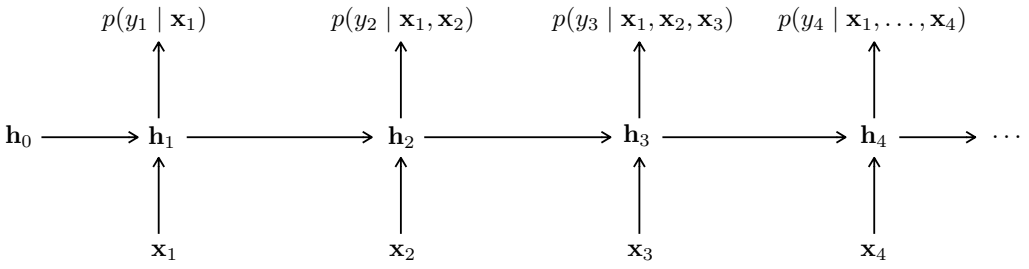


Figure 2.3: **Example RNN-based discriminative model.** Here, each time step is associated with exactly one input, $\mathbf{x}_t$, and one output label, y_t. Notice that each distribution over y_t is conditioned not only on the current input $\mathbf{x}_t$ but also on all previous inputs.

2.2.1 Discriminative Sequence Models

We have already seen one simple discriminative model above, where we modeled the probability of malignancy given a sequence of patient measurements, $p(y_T \mid \mathbf{x}_1, \ldots, \mathbf{x}_T)$. Another possible application of RNNs is to discriminatively model the distribution over a sequence of outputs, $y_1, \ldots, y_T$, given a sequence of inputs, $\mathbf{x}_1, \ldots, \mathbf{x}_T$, under the assumption that the outputs are conditionally independent *given the input sequence*. For example, one can consider surgical activity recognition from hand-movement data during robot-assisted surgery, where for each time step we aim to predict the current activity (e.g. suturing or knot tying), as will be discussed in more detail in Chapter 3. This is either done by conditioning on past motion data only (in the online setting) or by conditioning on both past and future motion data (in the offline setting).

In the online setting, we are restricted to leveraging only past and current inputs. Here, we model $p(y_t \mid \mathbf{x}_1, \ldots, \mathbf{x}_t)$ for all t. This model can be realized with RNNs

(again assuming binary classification for simplicity) via

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t, \boldsymbol{\theta}_h) \tag{2.26}$$

$$p(y_t \mid \mathbf{x}_1, \ldots, \mathbf{x}_t) = \text{softmax}(\mathbf{W}_{hy}\mathbf{h}_t + \mathbf{b}_y) \tag{2.27}$$

which is illustrated in Figure 2.3. This is also useful as a simple baseline even when online inference is not mandatory.

In the offline setting, we can also make use of *future* inputs, and this is done using *bidirectional RNNs*. The idea is simple: in order to leverage both past and future inputs, one RNN is run in the forward direction, and another is is run in the reverse direction, after which hidden states are concatenated and finally used together for prediction:

$$\mathbf{h}_t^{(\rightarrow)} = \mathbf{f}(\mathbf{h}_{t-1}, \mathbf{x}_t, \boldsymbol{\theta}_h^{(\rightarrow)}) \quad \text{for} \quad t = 1, 2, \ldots, T \tag{2.28}$$

$$\mathbf{h}_t^{(\leftarrow)} = \mathbf{f}(\mathbf{h}_{t+1}, \mathbf{x}_t, \boldsymbol{\theta}_h^{(\leftarrow)}) \quad \text{for} \quad t = T, T-1, \ldots, 1 \tag{2.29}$$

$$\mathbf{h}_t = [\mathbf{h}_t^{(\rightarrow)}; \mathbf{h}_t^{(\leftarrow)}] \tag{2.30}$$

$$p(y_t \mid \mathbf{x}_1, \ldots, \mathbf{x}_T) = \text{softmax}(\mathbf{W}_{hy}\mathbf{h}_t + \mathbf{b}_y) \tag{2.31}$$

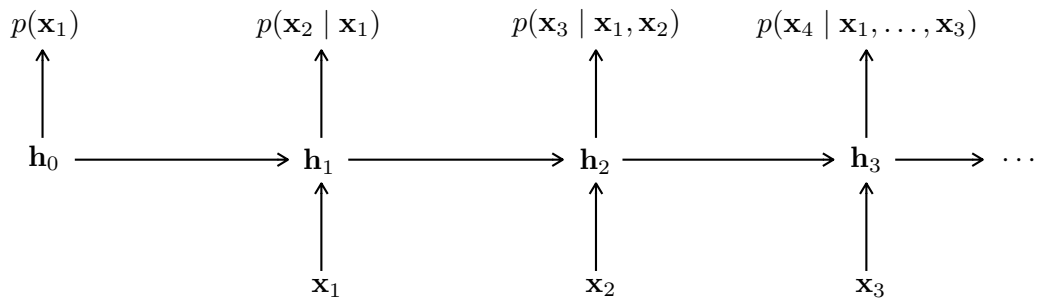


Figure 2.4: **Example RNN-based generative model.** At training time, each $\mathbf{x}_t$ fed into the model is taken directly from the observed sequence; this is known as *teacher forcing*. In contrast, at inference time, each input is *sampled* from the distribution governed by the previous time step.

2.2.2 Generative Sequence Models

Another use of RNNs is autoregressive density estimation, in which we model a full joint distribution by making use of the factorized form

$$p(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T) = p(\mathbf{x}_1)p(\mathbf{x}_2 \mid \mathbf{x}_1) \cdots p(\mathbf{x}_T \mid \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{T-1}) \tag{2.32}$$

Notice that the factorization above does *not* make any independence assumptions over time; this model has the potential to generate full trajectories of surgical motion through intermediate (learned) representations. This formulation is extremely general, in that we have not yet associated a parameterized distribution which each term in the product above. In Chapters 5 and 5, we will consider continuous distributions, but for simplicity, in this section we will assume that each time step is associated with a categorical distribution. Then, with each RNN step associated with one factor

in the joint distribution, we have

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_{t-1}, \boldsymbol{\theta}_h) \tag{2.33}$$

$$p(\mathbf{x}_t \mid \mathbf{x}_1, \ldots, \mathbf{x}_{t-1}) = \text{softmax}(\mathbf{W}_{hy}\mathbf{h}_t + \mathbf{b}_y) \tag{2.34}$$

This is illustrated in Figure 2.4.

2.2.3 Teacher Forcing

When training RNN-based generative models using the factorized representation $p(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T) = p(\mathbf{x}_1)p(\mathbf{x}_2 \mid \mathbf{x}_1) \cdots p(\mathbf{x}_T \mid \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{T-1})$, we typically have access to only a limited training dataset which consists of a fixed number of training sequences. For example, in the case of modeling surgical hand motion, imagine that the beginning of a sequence consists of a surgeon beginning to suture. The true underlying distribution captures many possible ways to complete the sequence (as there are many possible ways that the suture will be completed); however we often have access to only one such future – the specific future that was observed in the sequence that we collected. Because of this, there is a disconnect between *training* an RNN-based generative model and *sampling* from an RNN-based generative model at inference time. At training time, we resort to *teacher forcing*: at time t, when computing $f(\mathbf{h}_{t-1}, \mathbf{x}_{t-1}, \boldsymbol{\theta})$, we do not sample $\mathbf{x}_{t-1}$ from our model, but rather use the observed $\mathbf{x}_{t-1}$ that comes directly from our training data.

Chapter 3

Recognizing Surgical Activities with Recurrent Neural Networks

In this chapter, we focus on the joint localization and classification of surgical activities from motion data, which is also known as activity *recognition* in the literature. We introduce recurrent neural networks for this task and show that they are capable of achieving state-of-the-art performance. The most important contribution in this chapter is the demonstration that *maneuver* recognition is possible: unlike the activity granularities considered in prior work, maneuvers are already part of training curricula, and are therefore already familiar to surgeons.

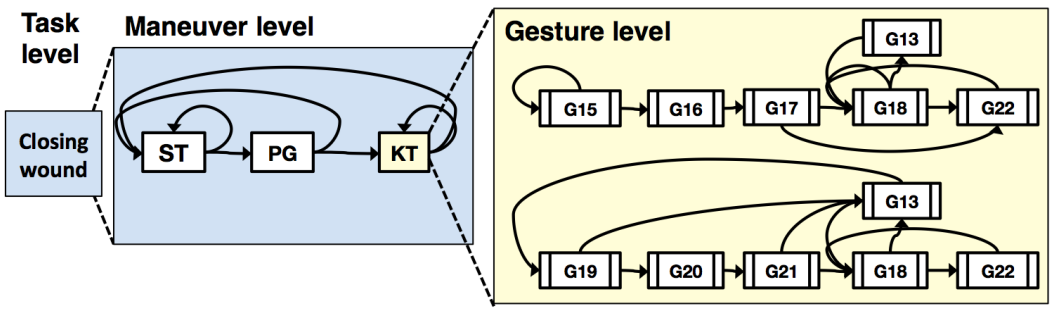


Figure 3.1: **Decomposition of the *closing wound* task into maneuvers, and of the *knot tying* maneuver into gestures.** Here the maneuvers are *suture throw* (ST), *pull grasp* (PG), and *knot tying* (KT); and the gestures are *grab suture using 2nd needle driver* (G13), *rotate suture twice using 1st needle driver around 2nd needle driver* (G15), *grab suture tail using 2nd needle driver in knot tying* (G16), *pull suture tail using 2nd needle driver through knot* (G17), *pull ends of suture taut* (G18), *rotate suture once using 2nd needle driver around 1st needle driver* (G19), *grab suture tail using 1st needle driver in knot tying* (G20), *pull suture tail using 1st needle driver through knot* (G21), and *grab suture using 1st needle driver* (G22).

3.1 Introduction

Automated surgical-activity recognition is a prerequisite to objective surgical-skill assessment and for providing targeted feedback to trainees. Previous research on automated surgical-activity recognition has focused on gestures within a surgical task [13–21]. Gestures are atomic segments of activity that typically last for a few seconds, such as grasping a needle. In contrast, maneuvers are composed of a sequence of gestures and represent higher-level segments of activity, such as tying a knot. Maneuvers exhibit more variability than the gestures from which they are composed, but they have the advantage of already being a part of current training curricula [7,8]. In addition, prior work focuses almost exclusively on hidden-Markov-model (HMM)

and conditional-random-field (CRF) based methods, with emphasis on pairwise and higher-order terms, which govern the probabilistic structure over activities. However, these prior works all define unary terms that depend only on inputs that are fairly local in time, as it is difficult to define unaries by hand which depend on distant inputs in a meaningful way.

In this chapter, we demonstrate the ability to recognize activities at the maneuver level for the first time; we focus on learning rich unary terms with recurrent neural networks; and we focus on the most difficult but realistic scenario, using the leave-one-user-out (LOUO) evaluation setup [19]. This last point emphasizes that fact that we wish to be able to generalize to new surgeons. In addition, we evaluate the effectiveness multiple recurrent neural networks (RNNs) for these tasks, assessing performance across 3 RNN architectures and assessing hyperparameter sensitivity for each, and in doing so we show that RNNs are capable of achieving state-of-the-art performance for both gesture and maneuver recognition.

Before proceeding, we specify our technical objective more precisely. While a surgeon or trainee operates a *da Vinci* surgical system, we have access to n_x kinematic signals over time. For example, for each hand, we might have positions, velocities, angular velocities, the gripper angle of the end effector, and so on. Let $\mathbf{x}_t \in \mathbb{R}^{n_x}$ refer to these measurements at a single time step t, so that $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$ denotes a kinematic sequence of length T. Our goal is surgical activity recognition, in which we map a sequence of kinematic signals $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$ to a sequence of activity labels

$y_1, y_2, \ldots, y_T$, with each label y_t being an integer that corresponds to one of n_y surgical activities. To accomplish this goal, it is common to proceed probabilistically, by discriminatively modeling $p(y_1, y_2, \ldots, y_T \mid \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$. This scenario is discussed at length in Chapter 2.

3.2 Prior Work

Most prior work has relied on hidden-Markov-model (HMM) [35] or conditional-random-field (CRF) [36, 37] based methods, including the latent-convolutional skip-chain CRF (LC-SC-CRF), which has achieved the lowest error rate for gesture recognition [13–17, 19, 20, 38]. In addition, recent work has also considered deep-learning based approaches, including temporal convolutional networks (TCNs) [18] and the combination of TCNs with reinforcement learning (TCNs+RL) in order to directly incorporate edit distance into the optimization process [21]. Because LC-SC-CRFs have outperformed all other methods with respect to error rate, we describe them briefly here, and we end by motivating the transition from LC-SC-CRFs to recurrent neural networks.

CRFs discriminatively model $p(y_1, y_2, \ldots, y_T \mid \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$, which (as opposed to HMMs) do not need to make naive assumptions about $p(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$. This is typically accomplished through an energy function consisting of unary terms (to capture label-input interactions) and pairwise terms (to capture label-label interac-

tions). LC-SC-CRFs enhance CRFs primarily in two ways: first, they add discrete latent variables at each time step (associated with unobserved subactivities); and second, they define pairwise terms over latent subactivities that are d time steps apart (rather than over adjacent subactivities). Looking forward, we put emphasis on the unary terms. For each latent subactivity, the unaries are restricted to be *local* in time and *linear* in model parameters, in turn deflecting nonlinear aspects of the model to be captured by the interaction between latent subactivities within first-order Markov chains. One could imagine enhancing this model by instead using k-th order Markov chains, however this is intractable because the cost of inference grows exponentially with k. We argue that this is overly restrictive, especially when modeling long, complex activities with non-linear dynamics, such as maneuvers, which can last 10s of seconds (see Figure 4.9). In contrast, recurrent neural networks are capable of modeling complex, nonlinear dynamics directly within the unary terms themselves, allowing for both rich representations and fast inference.

3.3 Methods

We evaluate three RNN architectures, simple RNNs, long short-term memory (LSTM), and gated recurrent units (GRUs), for modeling $p(y_1, y_2, \ldots, y_T \mid \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$, the distribution over surgical activities conditioned on observed motion. These architectures are described at length in Section 2.1, and discriminative modeling with

RNNs is described in Section 2.2.1. Here we review the main ideas within the context of surgical activity recognition.

We will use RNNs to model $p(y_1, y_2, \ldots, y_T \mid \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$, where again $y_1, y_2, \ldots, y_T$ is a sequence of activity labels and $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$ is a sequence of observed kinematics. First an RNN (specific variants described in Chapter 2) transforms the input sequence, according to its parameters, into a sequence of hidden states $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_T$, which can be interpreted (after training) as learned, task-specific representations of the input sequence. Next, each $\mathbf{h}_t$ is used to compute $p(y_t \mid \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$ via $\hat{\mathbf{p}}_t = \text{softmax}(\mathbf{W}_y \mathbf{h}_t + \mathbf{b}_y)$, where the i-th entry of $\hat{\mathbf{p}}_t$ is the estimated probability of the i-th activity and where $\mathbf{W}_y$, $\mathbf{b}_y$ are also parameters of the model. The activity labels over time are assumed to be conditionally independent given the input sequence, and training is carried out by maximizing conditional likelihood under this model, or equivalently by minimizing cross entropy.

Each particular architecture can also potentially benefit from multiple layers, bidirectionality, and regularization. Multiple layers are formed by letting the output of one RNN act as the input to another. In addition, in examining the RNN architectures above, one can immediately see that any particular hidden state $\mathbf{h}_t$ is computed using inputs *up to the current time step* t. Following nearly all previous work in gesture recognition, we focus on offline inference, for which our current predictions also depend on *future* time steps [13–21, 38]. This is achieved with bidirectional RNNs [39], which run one RNN in the forward direction, and one in the reverse, and

then concatenate the hidden states at each time step to form final hidden states. Finally, for regularization, we apply dropout [40] to the inputs $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$ and to $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_T$, the states emitted from the final RNN layer.

3.4 Datasets

For maneuver recognition, we use the Minimally Invasive Surgical Training and Innovation Center – Science of Learning (MISTIC-SL) dataset [41], and for gesture recognition, we use the JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS) dataset [19, 42]. Both datasets contain kinematic data collected from a *da Vinci* surgical system, along with activity annotations over time, provided manually by experts. In both cases, we use 14 kinematic signals as input: velocities, angular velocities, and gripper angle, all for both the left and right hands. Figure 3.2 shows frames over time that are representative of both MISTIC-SL and JIGSAWS (taken from MISTIC-SL).

MISTIC-SL contains annotations at the maneuver level and not at the gesture level, with each activity over time specified as *suture throw* (ST), *knot tying* (KT), *grasp pull run suture* (GPRS), or *intermaneuver segment* (IMS). We follow [42] and use a subset of 39 right-handed trials, performed by 15 subjects in total, for all experiments. These subjects were primarily residents, with varying levels of experience in robot-assisted surgery. Original signals are provided at 50Hz, which we downsample

by a factor of 6, as in [38]. The data was collected in a benchtop training environment, with each trial ideally consisting of a suture throw, a surgeon's knot, eight more suture throws, and a final surgeon's knot, though subjects do depart from this ideal sequence.

JIGSAWS contains annotations at the gesture level and not at the maneuver level. We follow the majority of prior work and consider the Suturing task, with each activity over time specified as one of 10 gestures, including *reaching for needle with right hand*, *pushing needle through tissue*, and so on (a full list can be found in [19]). Original signals are provided at 30Hz, which we first downsample by a factor of 6, as in [38]. The dataset is composed of 39 trials, performed by 8 subjects in total, and was also collected in a benchtop training environment. The 8 subjects were primarily medical students, with varying levels of experience in robot-assisted surgery (4 subjects reported fewer than 10 hours of experience; 2 subjects reported between 10 and 100 hours; and 2 subjects reported over 100 hours).

3.5 Experimental Design

The performance metrics used are frame-wise error rate (the percentage of incorrectly-labeled frames) and segment-level edit distance (or Levenshtein distance, the number of operations required to transform a segment-level prediction sequence into a segment-level ground-truth sequence). We remark that, although not necessary, it is

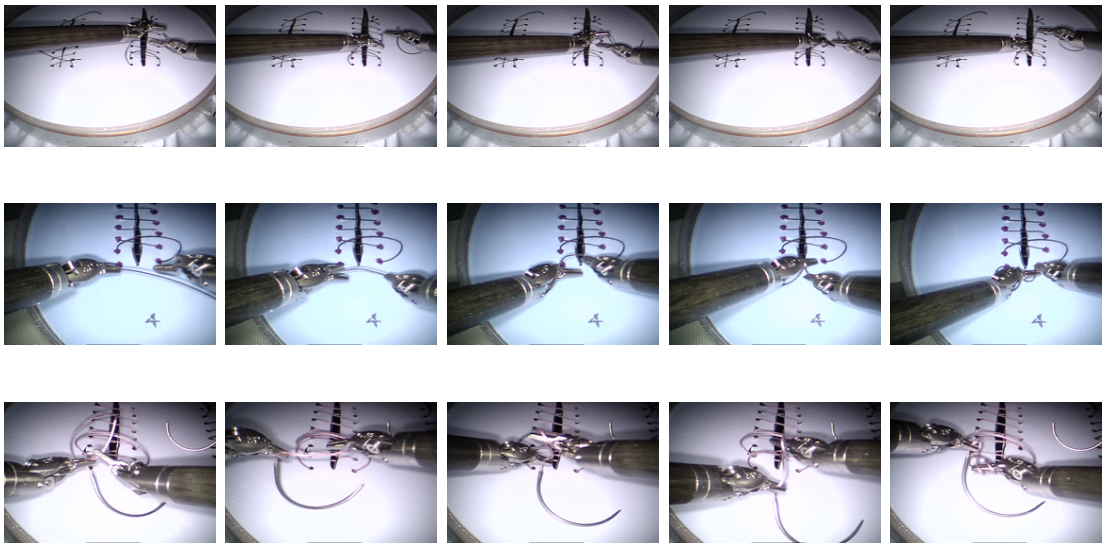


Figure 3.2: **Three example subsequences of knot tying from the MISTIC-SL dataset (1 Hz).**

common to normalize edit distance, either at the dataset level or the sequence level, to aid interpretability. As summarized in Table 3.3, both normalizations have been used in literature. For global normalization, the original edit distances are normalized by the maximum number of segments in any ground-truth sequence, here 59 for MISTIC-SL and 37 for JIGSAWS. For sequence-level normalization, the edit distance between each ground-truth, prediction pair is normalized by the maximum number of segments in *either* the ground-truth or predicted sequence. We report results using both normalizations for comparison with prior work. We advocate the use of the dataset-level normalization in future work for the following reasons. The dataset-level normalization provides immediate connections to the original edit distances, which aids analysis in future work; in contrast, one cannot invert the sequence-level normalizations without access to the number of segments in each predicted sequence. Sec-

ond, the dataset-level normalization continues to penalize predicted sequences with the same weight as more spurious segments are added; in contrast, the sequence-level normalization penalizes predicted sequences less and less as spurious segments are added.

To assess hyperparameter importance and select final hyperparameters, we use only the MISTIC-SL dataset. This is because the JIGSAWS dataset has predefined leave-one-user-out test folds with every user included in the test set [19]; and performing hyperparameter sweeps directly on test folds is questionable practice that may inflate our estimated ability to generalize to new data. We define a MISTIC-SL validation set using 4 users (those with one trial each), leaving 11 users for final performance assessment.

For hyperparameter analysis, we rely on the functional ANOVA (fANOVA) framework [43]. fANOVA lets us estimate marginal performance with respect to each hyperparameter, with all others marginalized out. Such marginalization is not trivial because integrals must be computed in sparsely-populated spaces. fANOVA offers a solution by a) approximating the mapping from hyperparameters to performance with a random forest and b) exploiting the random forest to approximate these integrals in linear time. To keep experiments tractable, we focus on 4 of the most relevant hyperparameters: the number of hidden units per layer, n_h, the number of layers, n_l, the shared dropout probability applied to the input kinematics and to the hidden states of the final layer, d, and the learning rate, α. Other design choices are

fixed for all experiments: we train for approximately 100 epochs (700 iterations with a batch size of 5); and we use the Adam optimizer with the decay rates for first and second moments fixed to their default values ($\beta_1 = 0.9, \beta_2 = 0.999$). For each RNN architecture, hyperparameters are explored using random search [44] with 200 trials. For each trial, $\log_2 n_h$ is chosen uniformly from $\{4, 5, \ldots, 9\}$; n_l is chosen uniformly from $\{1, 2, 3\}$; d is chosen uniformly from $[0, 0.5]$; and $\log_{10} \alpha$ is chosen uniformly from $[-4, -2]$.

As highlighted in the introduction, at test time, our objective is to generalize to a new sample of surgeons. With this goal in mind, to assess final test performance, we use the leave-one-user-out evaluation setup for both MISTIC-SL and JIGSAWS [19]: during the i-th run, all trials from the i-th user are left out of the training set and used as the test set.

3.6 Results

Here we provide an overview of results, first during the validation phase, in order to assess hyperparameter importance, and then during the test phase, in which a single set of hyperparameters for each architecture is chosen and evaluated in an exhaustive leave-one-user-out validation setup.

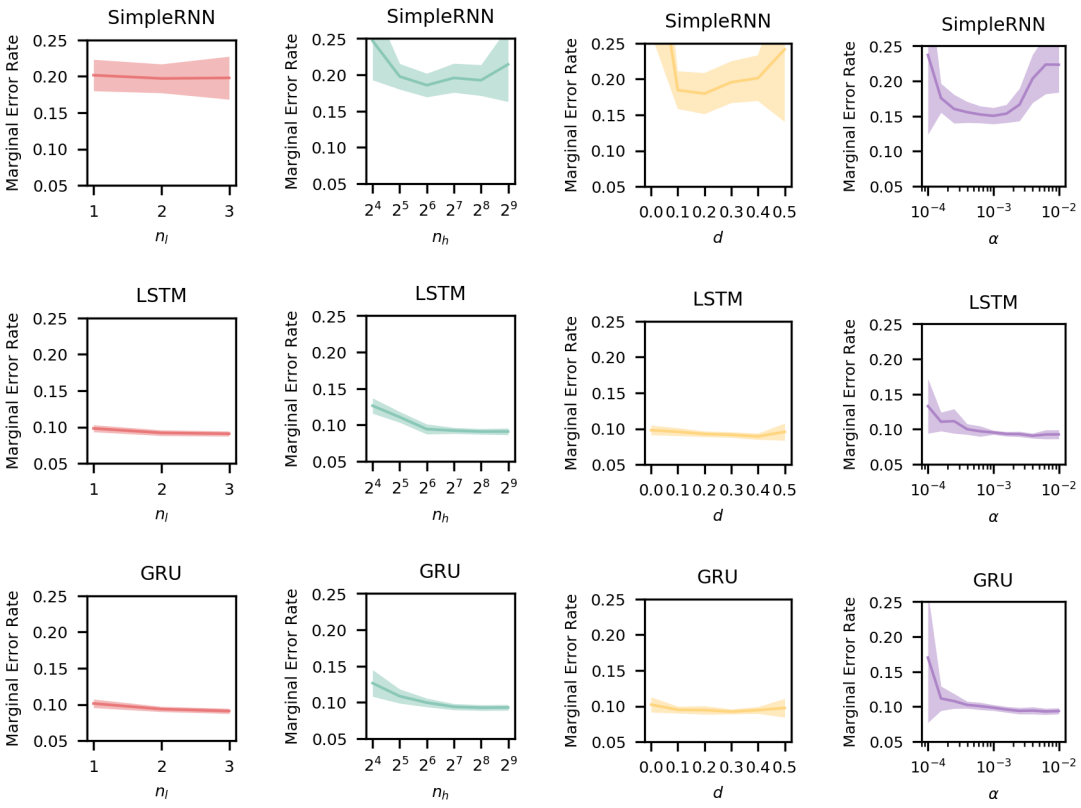


Figure 3.3: **Estimated marginal error rate for each hyperparameter (shown separately for each RNN architecture).** The dark line represents the mean and the top and bottom edges represent one standard deviation. From left to right, the hyperparameters are the number of layers n_l, the number of hidden units n_h, the dropout probability d, and the learning rate α.

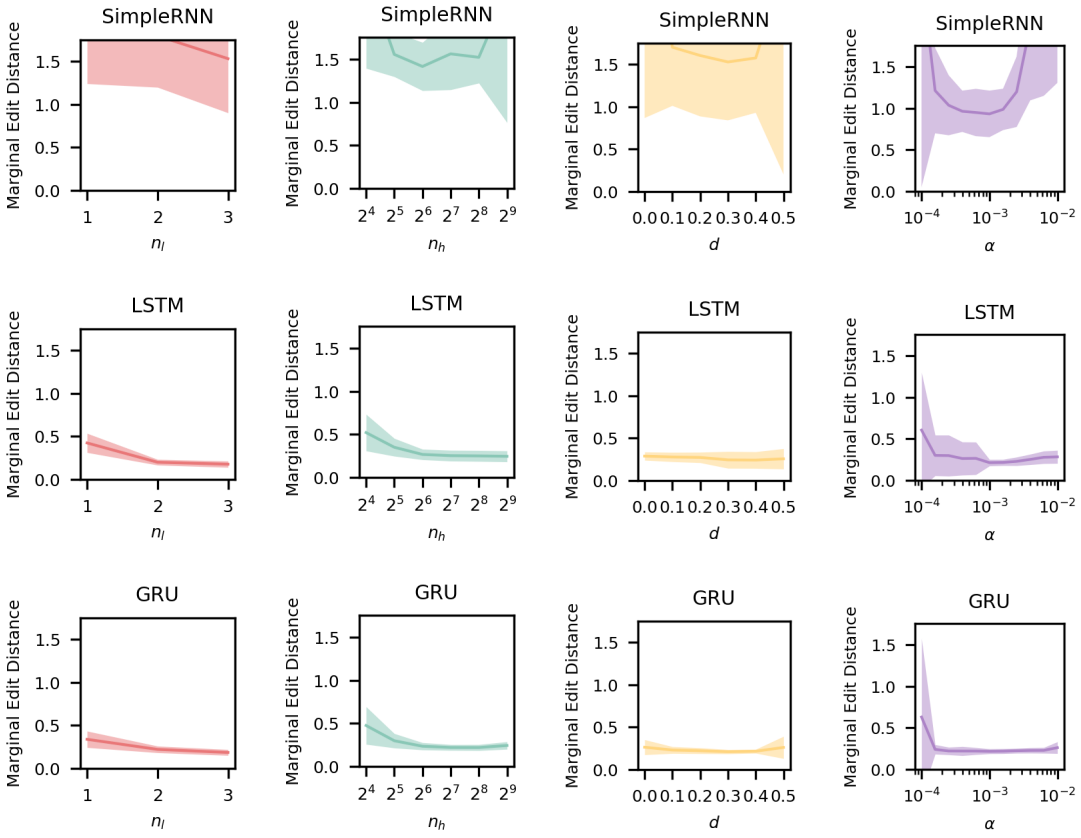


Figure 3.4: **Estimated marginal edit distance for each hyperparameter (shown separately for each RNN architecture).** The dark line represents the mean and the top and bottom edges represent one standard deviation. From left to right, the hyperparameters are the number of layers n_l, the number of hidden units n_h, the dropout probability d, and the learning rate α.

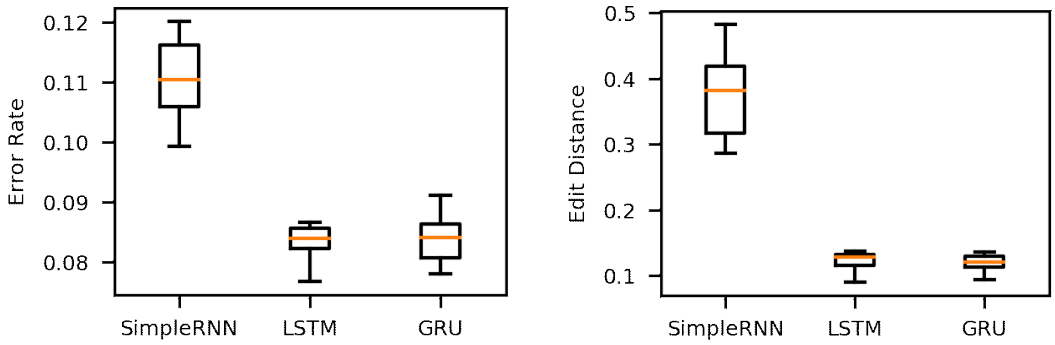


Figure 3.5: **Box plots summarizing error rates (left) and normalized edit distances (right) obtained with each RNN architecture, computed from the top 10% of validation runs.** Each box represents the 25th, 50th, and 75th percentiles, while the whiskers represent minimum and maximum values.

3.6.1 Hyperparameter Analysis and Validation-Set Performance

Figure 3.3 shows estimated marginal error rate for each hyperparameter (with each hyperparameter analyzed separately for each RNN architecture). The dark line represents mean error rate and the lighter areas indicate one standard deviation. Simple RNN mean error rates range between approximately 15% and 25%, while LSTM and GRU mean error rates range between approximately 10% and 15%. For these three architectures, we see similar behavior with respect to hyperparameters: the number of hidden units and the learning rate have the largest effects, and as either of these hyperparameters increase, the mean error rate tends to decrease approximately monotonically, until flattening off at approximately 64 or 128 hidden units or at a learning rate of approximately 10^{-3}.

Figure 3.4 shows estimated marginal edit distance for each hyperparameter, again where the dark line represents mean error rate and the lighter areas indicate one standard deviation. Simple RNN mean edit distances range approximately between 100% and 200%, while LSTM and GRU mean edit distances are between 20% and 50%. As in the case of error rate, increases in hidden unit counts tend to improve performance, up to a threshold of approximately 64 or 128 hidden units. In addition, for all architectures, as the number of layers increases, edit distance decreases.

Figure 3.5 shows box plots using the top 10% of validation runs for each architecture. In terms of error rate, the top-performing simple RNN models all achieve approximately 11%, whereas LSTM and GRUs achieve approximately 8%. In terms of edit distance, the top-performing simple RNN architecture achieves approximately 35%, whereas the top-performing LSTM and GRU models achieve approximately 10%.

Table 3.1 shows the final single set of hyperparameters used to assess final test-set performance for both MISTIC-SL (maneuver recognition) and JIGSAWS (gesture recognition), chosen to yield the lowest error rate over MISTIC-SL validation runs. This table also includes parameter counts and MISTIC-SL validation performance for these final models.

Table 3.1: **Final hyperparameters for each architecture. A single set of hyperparameters is used to evaluate test performance for both maneuver recognition (MISTIC-SL) and gesture recognition (JIGSAWS).** The hyperparameters are the number of layers, n_l, the number of hidden units per layer, n_h, the dropout probability, d, and the learning rate, α, chosen by minimizing error rate on the MISTIC-SL validation set.

	SimpleRNN	LSTM	GRU
n_l	2	3	3
n_h	512	64	128
d	0.3	0.4	0.3
α	$10^{-3.4}$	$10^{-2.5}$	$10^{-3.3}$
MISTIC-SL Val. Error Rate	11.1%	7.4%	7.8%
MISTIC-SL Val. Edit Dist.	35.6%	8.9%	9.3%
# Parameters	390k	60k	170k

Table 3.2: **MISTIC-SL mean error rates and normalized edit distances alongside results from prior work, sorted by error rate.** All results are averaged over users in a leave-one-user-out evaluation setup. Norm. Edit Dist.* uses the alternative sequence-level normalization described in Section 3.5.

	Error Rate (%)	Norm. Edit Dist. (%)	Norm. Edit Dist.* (%)
LC-SC-CRF [17]	18.3	29.7	—
Bidir SimpleRNN	11.6	31.7	26.9
Bidir LSTM	8.7	12.1	12.7
Bidir GRU	8.6	9.3	10.0

Table 3.3: **JIGSAWS mean error rates and normalized edit distances alongside results from prior work, sorted by error rate.** All results are averaged over users in a leave-one-user-out evaluation setup. Norm. Edit Dist.* uses the alternative sequence-level normalization described in Section 3.5.

	Error Rate (%)	Norm. Edit Dist. (%)	Norm. Edit Dist.* (%)
(Prior Work)			
MsM-CRF [14, 19]	32.2	—	—
KSVD-SHMM [13, 19]	26.6	—	—
GMM-HMM [19]	26.1	—	—
SDL + SC-CRF [20]	21.8	—	42.0
SDSDL [15]	21.3	—	16.7
TCN [18]	20.4	—	14.2
SC-CRF [16]	19.7	—	—
TCNs+RL [21]	17.9	—	12.1
LC-SC-CRF [17, 38]	16.6	—	23.1
(This Work)			
Bidir SimpleRNN	17.9	17.3	21.1
Bidir LSTM	15.3	8.4	11.9
Bidir GRU	15.2	8.4	11.5

3.6.2 Test-Set Performance

Tables 3.2 and 3.3 summarize our final test-set results for both maneuver recognition and gesture recognition in the context of prior work, where as a reminder we include 2 distinct forms of edit-distance normalization for comparison purposes. In the appendix, we also include a full summary of error rates and edit distances across all 11 test folds for MISTIC-SL and all 8 test folds for JIGSAWS (Tables 3.4 through 3.7). In the case of maneuver recognition, RNN based methods, and in particular GRUs, achieve an error rate of $8.6\% \pm 3.4\%$ and a normalized edit distance of $9.3\% \pm 4.3\%$. For comparison, LC-SC-CRFs achieve an error rate of 18.3% and an edit distance of 29.7%. In the case of gesture recognition, RNN based methods also lead to state-of-the-art performance, with LSTM and GRUs both achieving an error rate of approximately $15.3\% \pm 6.0\%$ and an edit distance of approximately $8.4\% \pm 6.0\%$ (and a sequence-level normalized edit distance of approximately $12.0\% \pm 7.0\%$). In previous work, LC-SC-CRFs were able to achieve an error rate of 16.6% (with a corresponding sequence-level normalized edit distance of 23.1%), and the TCNs+RL method was able to achieve a sequence-level normalized edit distance of 12.1% (with a corresponding error rate of 17.9%).

Figure 4.7 shows qualitative results for maneuver recognition. From top to bottom, the error rates are 20.3% for simple RNNs, 9.9% for LSTM, and 8.7% for GRUs, with normalized edit distances of 62.7%, 10.2%, and 3.4%. Here a single trial that is as representative as possible is shown for comparison purposes, but we note that no

single trial accurately represents all architectures. Additional qualitative results are shown on a per-trial basis in the appendix, showing individual trials with median error rate for each architecture (Figure 4.9) and individual trials with median edit distance for each architecture (Figure 4.10).

Figure 4.8 shows qualitative results for gesture recognition. From top to bottom, the error rates are 17.3% for simple RNNs, 12.9% for LSTM, and 11.9% for GRUs, with normalized edit distances of 18.9%, 5.4%, and 13.5%. Again we note that a single trial that is as representative as possible is shown, but that no single trial accurately represents all architectures. Additional qualitative results are shown on a per-trial basis in the appendix, showing individual trials with median error rate for each architecture (Figure 4.11) and individual trials with median edit distance for each architecture (Figure 4.12).

3.7 Discussion

Many interesting observations are worth highlighting in the context of the hyperparameter analysis. First, the number of hidden units per layer and the learning rate are both crucial hyperparameters. In particular, we emphasize that low learning rates that are sometimes fixed by pracitioners without experimentation, such as 10^{-4}, may lead to subpar (and unstable) performance, in terms of both error rate and edit distance. We remark that this is not because these trials were not run for enough

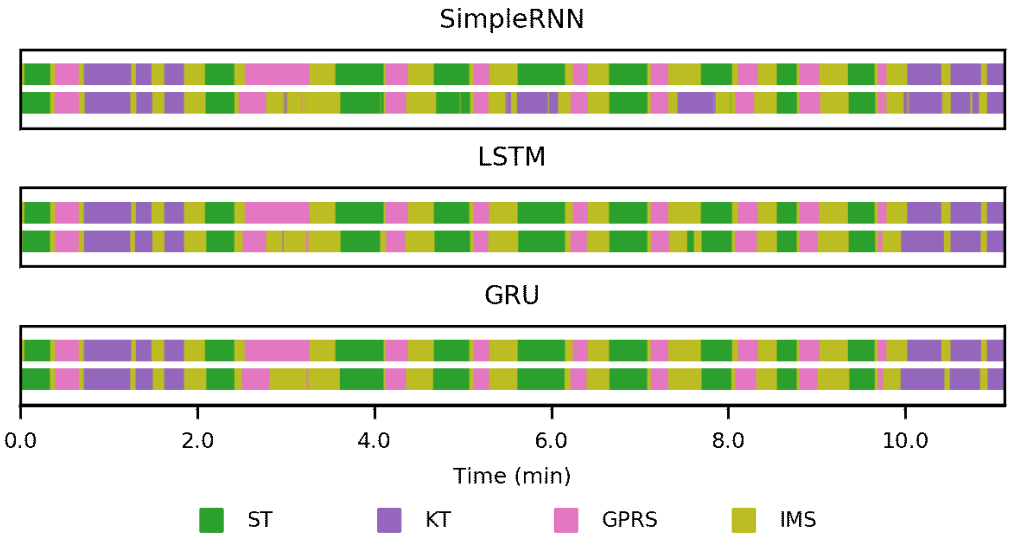


Figure 3.6: **Qualitative MISTIC-SL Results (Maneuver Recognition).** Ground truth labels are shown directly above predicted labels for each architecture. A single trial that is as representative as possible is shown to ease comparison; see the appendix for more results.

iterations: error rates and edit distances both typically stabilize early on in training, e.g. around epoch 20 of 100. Second, we can see that the number of RNN layers has only a minor impact on error rate, but has a significant impact on edit distance, with more layers yielding better (lower) edit distance.

An interesting though not particularly surprising result is that simple RNNs yield subpar error rates and subpar edit distances in comparison to other RNN architectures. Here, two key differences between simple RNNs and LSTM/GRUs are 1. vulnerability to the vanishing gradient problem, or in other words the ability to learn long-term dependencies from data, and 2. an implicit bias toward smooth predictions which is present in LSTM/GRUs but not in simple RNNs. LSTM and GRUs, which both alleviate the vanishing gradient problem and maintain a bias toward smooth

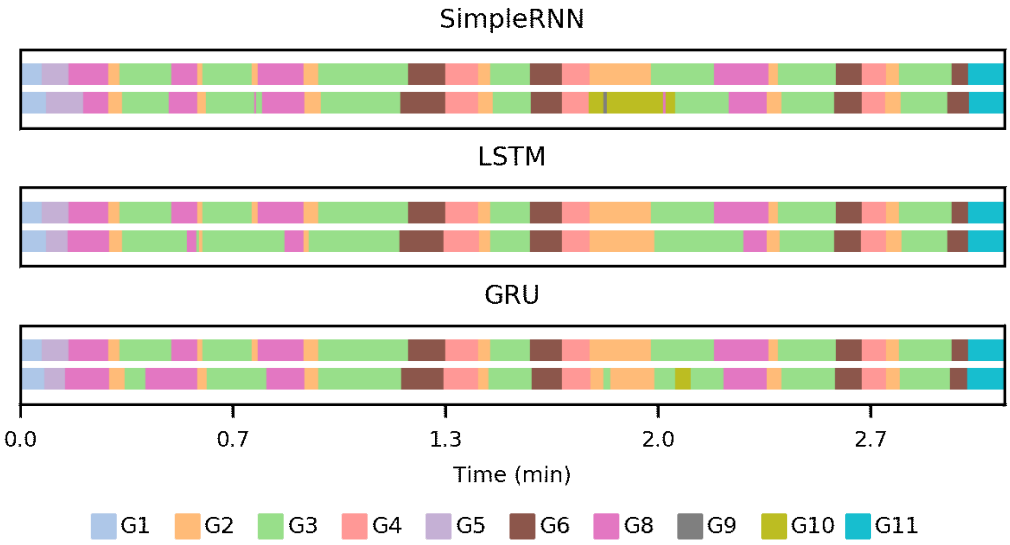


Figure 3.7: **Qualitative JIGSAWS Results (Gesture Recognition).** Ground truth labels are shown directly above predicted labels for each architecture. A single trial that is as representative as possible is shown to ease comparison; see the appendix for more results.

predictions, obtain both low error rates and low edit distances.

With respect to prior work, we can see that RNNs are quite competitive (Tables 3.2 and 3.3). For maneuver recognition, RNNs lead to substantial gains over the best non-RNN based method (the LC-SC-CRF), more than halving the error rate (8.6% vs. 18.3%) and edit distance (9.3% vs. 29.7%). For gesture recognition, RNNs also lead to state-of-the-art peformance, achieving a best error error rate of 15.2% (vs. 16.6% for the best non-RNN based method, the LC-SC-CRF), and a best edit distance of 8.4% (or 11.5% in terms of the alternative edit distance definition that is directly comparable to that of the best non-RNN based method, referred to here as TCNs+RL, which achieve 12.1%). The latter result is especially interesting because

in [21] reinforcement learning was leveraged specifically so that edit distance could be incorporated into the optimization process.

We note that care should be taken when interpreting these tables, and others in literature, especially in the case of JIGSAWS. All results shown here use the leave-one-user-out evaluation setup, but they nevertheless vary in other experimental aspects: 1) In some cases, only a small subset of kinematics is used (as in this work), whereas in some cases all 76 kinematic signals are used, while in others the inputs are never explicitly specified. 2) In some cases, downsampling is performed (as in this work), while in other cases the full temporal resolution is maintained. 3) In some cases, hyperparameters are chosen independently of the test folds (as in this work), whereas in other cases hyperparameters are tuned directly on the test folds. 4) In most cases, results are indicative of *offline* operation (as in this work), whereas in some cases results are indicative of *online* operation, during which predictions cannot not depend on future inputs.

Finally, we remark that although all experiments here pertain to offline operation, it is not difficult to leverage RNNs for online operation. To remain causal, one can use unidirectional RNNs rather than bidirectional RNNs. All of the architectures considered here are efficient enough to easily run in real time, provided access to a GPU.

3.8 Conclusions

The major conclusion of this chapter is that maneuver recognition is feasible with RNNs, despite maneuvers exhibiting more complexity and variability than the gestures from which they are composed. This is an exciting result because it offers the opportunity to provide targeted assessment and feedback at a higher level of granularity, and one that is already part of existing training curricula. In addition, we found that RNNs yield state-of-the-art performance, without any probabilistic structure over activities, because of their ability to learn rich, non-linear unary terms from data. We provided results for 3 RNN architectures and an analysis of hyperparameter sensitivity for each, in order to ease the adoption of RNNs in future work. Finally, we obtained state-of-the-art performance not only for maneuver recognition but also for gesture recognition, using hyperparameters that were carried over from maneuver recognition. This suggests that these hyperparameters lead to fairly robust performance across these related tasks.

In moving forward, we highlight two limitations of this section: 1. Although we know that LSTM alleviates the vanishing gradient problem, we also know that it in no way *solves* the problem, and in practice it is difficult to learn dependencies beyond 10s of time steps. Can alleviating the vanishing gradient problem further lead to improved performance, especially for maneuver recognition, in which activities span long time scales? 2. For both gesture recognition and maneuver recognition, we required many densely-annotated sequences in order to train our models; and

collecting these annotations is tedious, expensive, and error-prone. Is it possible to make use of surgical motion data in the absence of annotations; and is activity recognition feasible when annotated data is scarce? These two questions are addressed in the chapters that follow.

3.9 Appendix

Table 3.4: **MISTIC-SL test-set error rates (%).** Each of the first 11 rows correspond to a separate training-and-evaluation run with one user left out for testing. The final row corresponds to means and standard deviations across users.

SimpleRNN	LSTM	GRU
14.4	9.1	**8.8**
11.4	12.2	11.6
10.3	7.0	**5.3**
7.0	5.7	**5.3**
7.7	**4.6**	6.4
20.3	**14.0**	16.1
8.3	**4.6**	4.8
8.5	6.6	**6.5**
13.5	**8.6**	8.8
11.9	**8.5**	8.8
14.6	15.2	**12.4**
11.6 ± 3.8	8.7 ± 3.5	8.6 ± 3.4

Table 3.5: **MISTIC-SL test-set edit distances (%).** Each of the first 11 rows correspond to a separate training-and-evaluation run with one user left out for testing. The final row corresponds to means and standard deviations across users.

SimpleRNN	LSTM	GRU
50.8	14.7	**7.9**
38.1	11.9	**6.8**
21.2	6.8	**3.4**
25.4	7.6	**5.1**
22.0	12.7	**11.9**
19.2	13.6	**5.6**
38.1	10.7	**7.9**
32.5	12.9	**12.2**
34.7	**14.4**	17.8
25.4	**7.2**	7.6
41.1	21.2	**15.7**
31.7 ± 9.5	12.1 ± 4.0	9.3 ± 4.3

Table 3.6: **JIGSAWS test-set error rates (%).** Each of the first 8 rows correspond to a separate training-and-evaluation run with one user left out for testing. The final row corresponds to means and standard deviations across users.

SimpleRNN	LSTM	GRU
14.1	17.0	14.6
12.6	**11.5**	12.2
34.1	29.8	**29.5**
18.2	15.1	**8.5**
12.3	**10.5**	12.1
22.1	18.8	**18.2**
15.3	**8.6**	12.7
14.2	**11.4**	14.3
17.9 ± 6.8	15.3 ± 6.4	15.2 ± 6.0

Table 3.7: **JIGSAWS test-set edit distances (%).** Each of the first 8 rows correspond to a separate training-and-evaluation run with one user left out for testing. The final row corresponds to means and standard deviations across users.

SimpleRNN	LSTM	GRU
13.0	**8.1**	9.2
5.4	**3.2**	3.8
26.5	17.3	**15.1**
8.6	9.2	**1.6**
6.5	**1.6**	2.2
35.1	**17.8**	21.1
17.6	**4.7**	6.1
25.9	**4.9**	8.1
17.3 ± 10.2	8.4 ± 5.8	8.4 ± 6.3

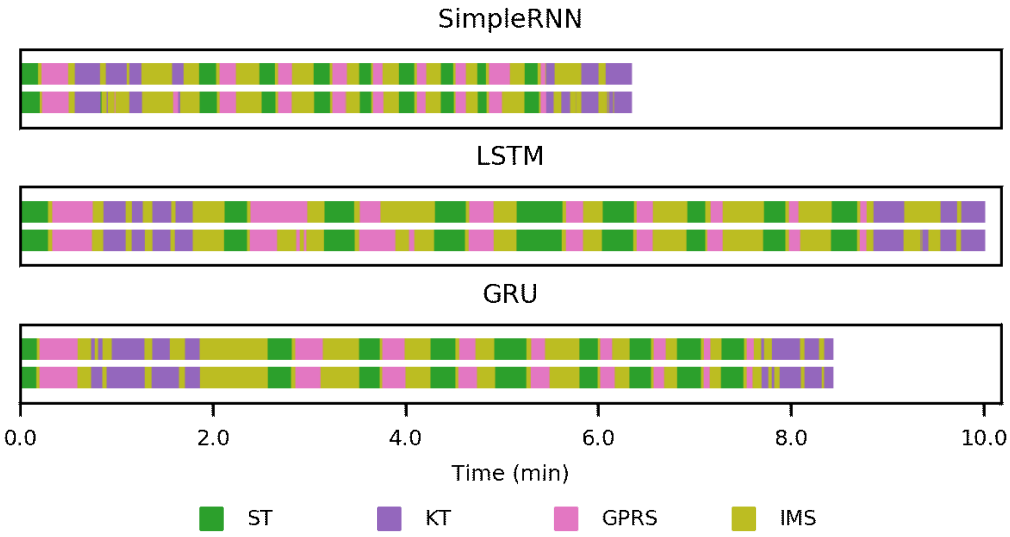


Figure 3.8: **Qualitative results: MISTIC-SL trials with median error rate (for each architecture).** Ground truth labels are shown above predicted labels. From top to bottom, the error rates are 11.3%, 7.7%, and 7.0% (with normalized edit distances of 30.5%, 20.3%, and 6.8%).

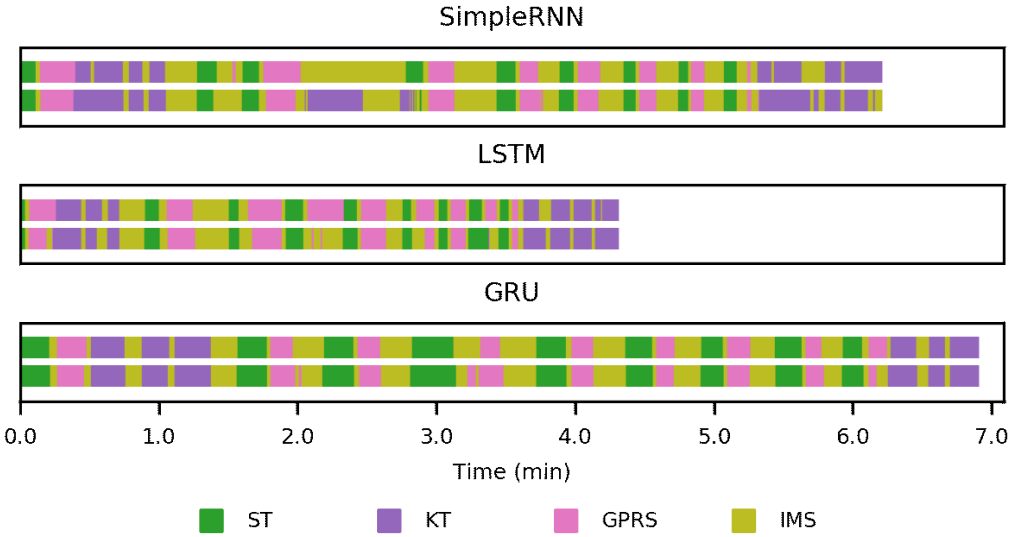


Figure 3.9: **Qualitative results: MISTIC-SL trials with median edit distance (for each architecture).** Ground truth labels are shown above predicted labels. From top to bottom, the normalized edit distances are 30.5%, 10.2%, and 6.8% (with error rates of 15.8%, 16.3%, and 6.4%).

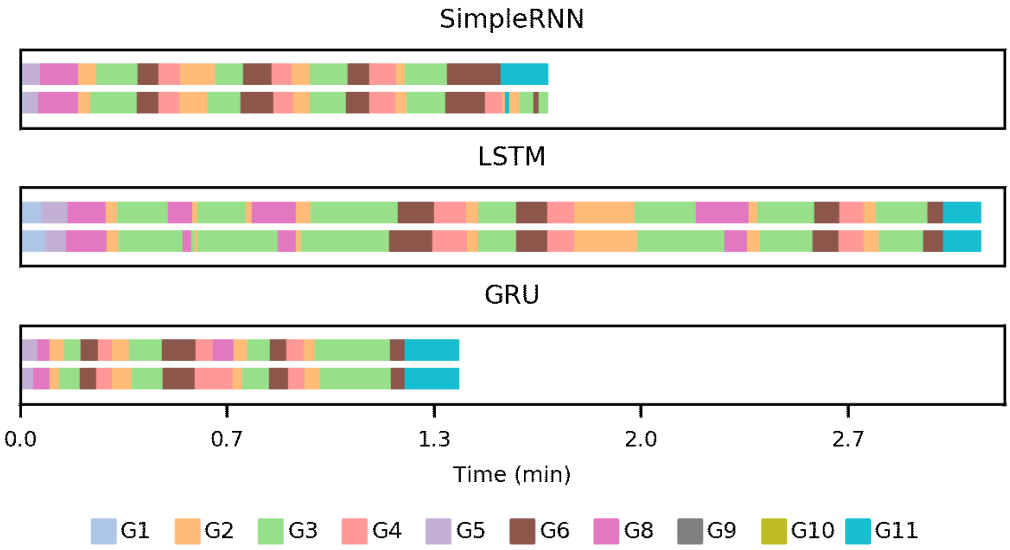


Figure 3.10: **Qualitative results: JIGSAWS trials with median error rate (for each architecture).** Ground truth labels are shown above predicted labels. From top to bottom, the error rates are 16.1%, 12.9%, and 12.3% (with normalized edit distances of 16.2%, 5.4%, and 2.7%).

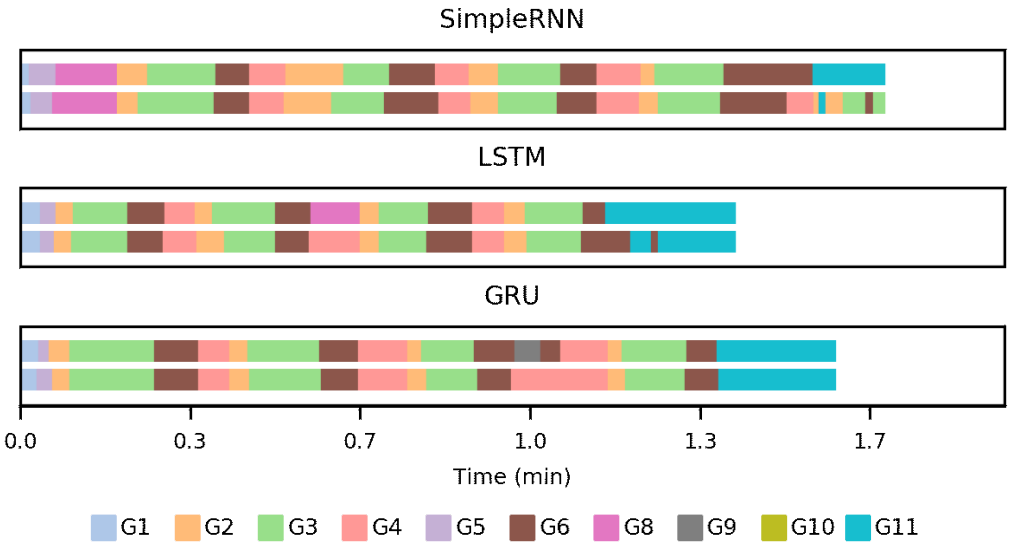


Figure 3.11: **Qualitative results: JIGSAWS trials with median edit distance (for each architecture).** Ground truth labels are shown above predicted labels. From top to bottom, the normalized edit distances are 16.2%, 8.1%, and 5.4% (with error rates of 16.1%, 15.0%, and 9.0%).

Chapter 4

Mixed History Recurrent Neural Networks for Learning Long-Term Dependencies

In the previous chapter we explored recurrent neural networks (RNNs) for the task of surgical activity recognition. An important question is whether performance is limited by the ability to learn long-term dependencies from data, a difficulty which has affected recurrent neural networks since their inception. In this section, we analyze the use of skip connections over time for learning long-term dependencies, both theoretically and empirically, and we introduce a new RNN architecture that is capable of learning dynamics over extremely long-term time periods. This architecture is evaluated alongside the most popular RNN architectures to date, both in the context

of surgical activity recognition and other tasks.

4.1 Introduction

Although recurrent neural networks have achieved state-of-the-art performance across many diverse tasks, from machine translation to surgical activity recognition, training them to capture long-term dependencies remains difficult. To date, the vast majority of successful RNN architectures alleviate this problem using nearly-additive connections between states, as introduced by long short-term memory (LSTM). Here we take an orthogonal approach and introduce mixed history Recurrent Neural Networks (MIST RNNs), a NARX RNN architecture that allows direct connections from the very distant past. We show that MIST RNNs 1) exhibit superior vanishing-gradient properties in comparison to LSTM and previously-proposed NARX RNNs; 2) are far more efficient than previously-proposed NARX RNN architectures, requiring even fewer computations than LSTM; and 3) improve performance substantially over LSTM and Clockwork RNNs on tasks requiring very long-term dependencies.

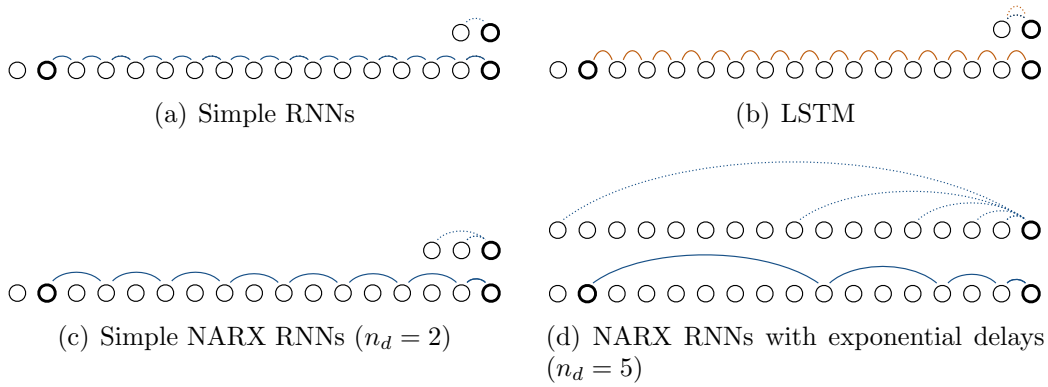
The vast majority of RNN successes rely on the mechanism introduced by long short-term memory [32, 45], which was intentionally designed to alleviate the so called *vanishing gradient problem* [30, 46]. The problem is that gradient contributions from events at time $t-\tau$ to a loss at time t diminish exponentially fast with τ, thus making it extremely difficult to learn from distant events (see Figures 4.1 and 4.2). LSTM alle-

viates the problem using nearly-additive connections between adjacent states, which help push the base of the exponential decay toward 1. However LSTM in no way solves the problem, and in many cases still fails to learn long-term dependencies (see, e.g., [47]).

NARX[1] RNNs [48] offer an orthogonal mechanism for dealing with the vanishing gradient problem, by allowing direct connections, or delays, from the distant past. However NARX RNNs have received much less attention in literature than LSTM, which we believe is for two reasons. First, as previously introduced, NARX RNNs have only a small effect on vanishing gradients, as they reduce the exponent of the decay by only a factor of n_d, the number of delays. Second, as previously introduced, NARX RNNs are extremely inefficient, as both parameter counts and computation counts grow by the same factor n_d.

In this chapter, we introduce MIST RNNs, a novel NARX RNN architecture which 1) exhibits superior vanishing-gradient properties in comparison to LSTM and previously-proposed NARX RNNs; 2) improves performance substantially over LSTM on tasks requiring very long-term dependencies; and 3) remains efficient in parameters and computation, requiring even fewer than LSTM for a fixed number of hidden units. Importantly, MIST RNNs reduce the decay's exponent by a factor of 2^{n_d-1}; see Figure 4.2.

[1]The acronym NARX stems from Nonlinear AutoRegressive models with eXogeneous inputs.



(a) Simple RNNs (b) LSTM

(c) Simple NARX RNNs ($n_d = 2$) (d) NARX RNNs with exponential delays ($n_d = 5$)

Figure 4.1: **Direct connections (dashed) to a single time step t and example shortest paths (solid) from time $t - \tau$ to time t for various architectures.** Typical RNN connections (blue) impede gradient flow through matrix multiplications and nonlinearities. LSTM facilitates gradient flow through additional paths between adjacent time steps with less resistance (orange). NARX RNNs facilitate gradient flow through additional paths that span multiple time steps.

4.2 Prior Work

Recurrent neural networks, as commonly described in literature, take on the general form

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t, \boldsymbol{\theta}) \tag{4.1}$$

which compute a new state $\mathbf{h}_t$ in terms of the previous state $\mathbf{h}_{t-1}$, the current input $\mathbf{x}_t$, and some parameters $\boldsymbol{\theta}$ (which are shared over time).

One of the earliest variants, now known to be especially vulnerable to the vanishing gradient problem, is that of simple RNNs [24], described by

$$\mathbf{h}_t = \tanh(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \mathbf{x}_t + \mathbf{b}) \tag{4.2}$$

In this equation and elsewhere in this paper, all weight matrices $\mathbf{W}$ and biases $\mathbf{b}$ collectively form the parameters $\boldsymbol{\theta}$ to be learned, and tanh is always written explicitly.

Long short-term memory [32,45], the most widely-used RNN architecture to date, was specifically introduced to address the vanishing gradient problem. As discussed in Chapter 2, the term LSTM is often overloaded; we refer to the variant with forget gates and without peephole connections. We reproduce the formulation for LSTM here,

$$\mathbf{f}_t = \sigma(\mathbf{W}_{fh}\mathbf{h}_{t-1} + \mathbf{W}_{fx}\mathbf{x}_t + \mathbf{b}_f) \tag{4.3}$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_{ih}\mathbf{h}_{t-1} + \mathbf{W}_{ix}\mathbf{x}_t + \mathbf{b}_i) \tag{4.4}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{oh}\mathbf{h}_{t-1} + \mathbf{W}_{ox}\mathbf{x}_t + \mathbf{b}_o) \tag{4.5}$$

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_{ch}\mathbf{h}_{t-1} + \mathbf{W}_{cx}\mathbf{x}_t + \mathbf{b}_c) \tag{4.6}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \tag{4.7}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{4.8}$$

with emphasis on Equation 4.7. This is the key update that allows LSTM to learn longer-term dependencies than simple RNNs. Discussed in this context, it is useful to interpret LSTM as 1. providing a simple-RNN based update, via Equation 4.6, but 2. updating an internal representation according to this simple RNN in a limited fashion, via Equation 4.7. In other words, rather than fully replacing the internal state at every time step, LSTM learns to update only when necessary. Looking forward, we will see

that this unimpeded flow across time holds not only for the forward pass but also for the backward pass, thus allowing gradient signals to flow back with less resistance. This is why LSTM is often initialized at the start of training so that forget gates are close to 1.0. Gated recurrent units [33] alleviate the vanishing gradient problem using this same idea.

NARX RNNs [48] also address the vanishing gradient problem, but using a mechanism that is orthogonal to (and possibly complementary to) that of LSTM. This is done by allowing *delays*, or direct connections from the past. NARX RNNs in their general form are described by

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{h}_{t-2}, \ldots, \mathbf{x}_t, \mathbf{x}_{t-1}, \ldots, \boldsymbol{\theta}) \tag{4.9}$$

but literature typically assumes the specific variant explored in [48],

$$\mathbf{h}_t = \tanh\left(\left[\sum_{d=1}^{n_d} \mathbf{W}_d \mathbf{h}_{t-d}\right] + \mathbf{W}_x \mathbf{x}_t + \mathbf{b}\right) \tag{4.10}$$

which we refer to as *simple* NARX RNNs.

Note that simple NARX RNNs require approximately n_d as much computation and n_d as many parameters as their simple-RNN counterpart (with $n_d = 1$), which greatly hinders their applicability in practice. To our knowledge, this drawback holds for all NARX RNN variants before MIST RNNs. For example, in [49], higher-order recurrent neural networks (HORNNs) are defined precisely as simple NARX RNNs,

and every variant in the paper suffers from this exact same problem. And, in [50], a simple NARX RNN architecture is defined that is limited to having precisely two delays with non-zero weights. This way, at the expense of having fewer, longer paths to the past, parameter and computation counts are only doubled.

The previous work that is most similar to ours, at least in spirit, is perhaps that of Clockwork RNNs [51], which split weights and hidden units into partitions, each with a distinct period. When it's not a partition's time to tick, its hidden units are passed through unchanged, and so Clockwork RNNs in some ways mimic NARX RNNs. However Clockwork RNNs differ in two key ways. First, Clockwork RNNs sever high-frequency-to-low-frequency paths, thus making it difficult to learn long-term behavior that must be detected at high frequency (for example, learning to depend on quick motions from the past for activity recognition). Second, Clockwork RNNs require hidden units to be partitioned *a priori*, which in practice is difficult to do in any meaningful way. NARX RNNs (and in particular MIST RNNs) suffer from neither of these drawbacks.

Many other approaches have also been proposed to capture long-term dependencies. Notable approaches include maintaining a generative model over inputs and learning to process only unexpected inputs [52], operating explicitly at multiple time scales [53], Hessian-free optimization [54], using associative or explicit memory [55–58], and initializing or restricting weight matrices to be orthogonal [47, 59].

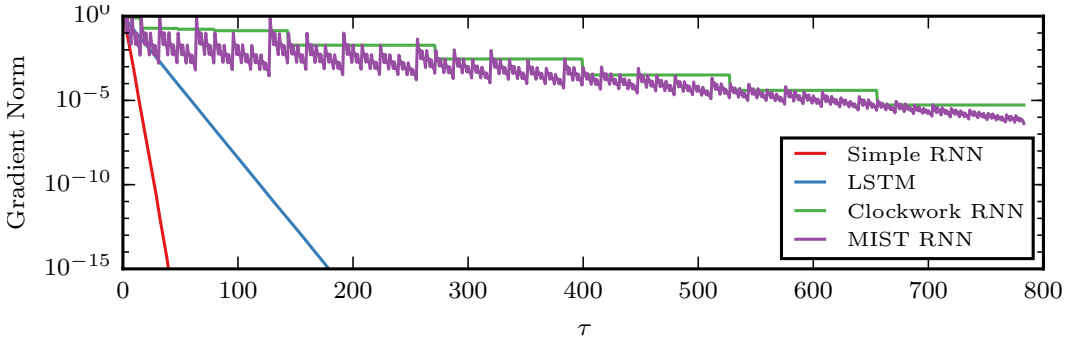


Figure 4.2: **Gradient norms $\|\frac{\partial^+ l_t}{\partial \mathbf{h}_{t-\tau}}\|$ averaged over a batch of examples during permuted MNIST training.** Unlike Clockwork RNNs and MIST RNNs, simple RNNs and LSTM capture essentially no learning signal from inputs that are far from the loss.

4.3 The Vanishing Gradient Problem in the Context of NARX RNNs

In [31, 46], gradient decompositions and sufficient conditions for vanishing gradients are presented for simple RNNs, which contain one path between times $t-\tau$ and t. Here, we use the *chain rule for ordered derivatives* [60] to connect gradient components to paths and edges, which in turn provides a simple extension of the results from [31, 46] to general NARX RNNs. We remark that we rely on slightly overloaded notation for clarity, as otherwise notation becomes cumbersome (see [61]).

We begin by disambiguating notation, as the symbol $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ is routinely overloaded in literature. Consider the Jacobian of $\mathbf{f}(\mathbf{x}, \mathbf{u}(\mathbf{x}))$ with respect to $\mathbf{x}$. We let $\frac{\partial^+ \mathbf{f}}{\partial \mathbf{x}}$ denote $\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}}$, a collection of full derivatives, and we let $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ denote $\frac{\partial \mathbf{f}(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}$, a collection of partial derivatives. This lets us write the ordinary chain rule as $\frac{\partial^+ \mathbf{f}}{\partial \mathbf{x}} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \frac{\partial^+ \mathbf{x}}{\partial \mathbf{x}} + \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \frac{\partial^+ \mathbf{u}}{\partial \mathbf{x}}$. Note

that this notation is consistent with [60,61], but is the exact opposite of the convention used in [31].

4.3.1 The Chain Rule for Ordered Derivatives

Consider an ordered system of n vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$, where each is a function of all previous:

$$\mathbf{v}_i \equiv \mathbf{v}_i(\mathbf{v}_{i-1}, \mathbf{v}_{i-2}, \ldots, \mathbf{v}_1), \quad 1 \leq i \leq n \tag{4.11}$$

The chain rule for ordered derivatives expresses the full derivatives $\frac{\partial^+ \mathbf{v}_i}{\partial \mathbf{v}_j}$ for any $j < i$ in terms of the full derivatives that relate $\mathbf{v}_i$ to all previous $\mathbf{v}_k$:

$$\frac{\partial^+ \mathbf{v}_i}{\partial \mathbf{v}_j} = \sum_{i \geq k > j} \frac{\partial^+ \mathbf{v}_i}{\partial \mathbf{v}_k} \frac{\partial \mathbf{v}_k}{\partial \mathbf{v}_j}, \quad j < i \tag{4.12}$$

4.3.2 Gradient Decomposition for General NARX RNNs

Consider NARX RNNs in their general form (Equation 4.9), which we remark encompasses other RNNs such as LSTM as special cases. Also, for simplicity, consider the situation that is most often encountered in practice, where the loss at time t is defined in terms of the current state $\mathbf{h}_t$ and its own parameters $\boldsymbol{\theta}_l$ (which are

independent of $\boldsymbol{\theta}$).

$$l_t = f_l(\mathbf{h}_t, \boldsymbol{\theta}_l) \tag{4.13}$$

(This is in not necessary, but we proceed this way to make the connection with RNNs in practice evident. For example, f_l may be a linear transformation with parameters $\boldsymbol{\theta}_l$ followed by squared-error loss.) Then the Jacobian (or transposed gradient) with respect to $\boldsymbol{\theta}$ can be written as

$$\frac{\partial^+ l_t}{\partial \boldsymbol{\theta}} = \frac{\partial f_l}{\partial \mathbf{h}_t} \frac{\partial^+ \mathbf{h}_t}{\partial \boldsymbol{\theta}} \tag{4.14}$$

because the additional term $\frac{\partial f_l}{\partial \boldsymbol{\theta}_l} \frac{\partial^+ \boldsymbol{\theta}_l}{\partial \boldsymbol{\theta}}$ is $\mathbf{0}$. Now, by letting $\mathbf{v}_1 = \boldsymbol{\theta}$, $\mathbf{v}_2 = \mathbf{x}_1$, $\mathbf{v}_3 = \mathbf{x}_2$, and so on in Equations 4.11 and 4.12, we immediately obtain

$$\frac{\partial^+ \mathbf{h}_t}{\partial \boldsymbol{\theta}} = \sum_{\tau=0}^{t-1} \frac{\partial^+ \mathbf{h}_t}{\partial \mathbf{h}_{t-\tau}} \frac{\partial \mathbf{h}_{t-\tau}}{\partial \boldsymbol{\theta}} \tag{4.15}$$

because all of the partials $\frac{\partial \mathbf{x}_{t-\tau}}{\partial \boldsymbol{\theta}}$ are $\mathbf{0}$.

Equations 4.14 and 4.15 extend Equations 3 and 4 of [31] to general NARX RNNs, which encompass simple RNNs, LSTM, etc., as special cases. This decomposition breaks $\frac{\partial^+ \mathbf{h}_t}{\partial \boldsymbol{\theta}}$ into its temporal components, making it clear that the spectral norm of $\frac{\partial^+ \mathbf{h}_t}{\partial \mathbf{h}_{t-\tau}}$ plays a major role in how $\mathbf{h}_{t-\tau}$ affects the final gradient $\frac{\partial^+ l_t}{\partial \boldsymbol{\theta}}^T$. In particular, if the norm of $\frac{\partial^+ \mathbf{h}_t}{\partial \mathbf{h}_{t-\tau}}$ is extremely small, then $\mathbf{h}_{t-\tau}$ has only a negligible effect on the final gradient, which in turn makes it extremely difficult to learn from events that

occurred at $t - \tau$.

4.3.3 Connecting Gradient Components to Paths and Edges

Equations 4.14 and 4.15, along with the chain rule for ordered derivatives, let us connect gradient components to paths and edges, which is useful for a) gaining insights into various architectures and b) solidifying intuitions from backpropagation through time which suggest that short paths between $t - \tau$ and t facilitate gradient flow.

We will apply Equation 4.12 repeatedly to associate gradient components with paths connecting $t - \tau$ to t, beginning with Equation 4.15 and handling simple RNNs and simple NARX RNNs in order. Applying Equation 4.12 to expand $\frac{\partial^+ \mathbf{h}_t}{\partial \mathbf{h}_{t-\tau}}$, we obtain

$$\frac{\partial^+ \mathbf{h}_t}{\partial \mathbf{h}_{t-\tau}} = \sum_{t \geq t' > t-\tau} \frac{\partial^+ \mathbf{h}_t}{\partial \mathbf{h}_{t'}} \frac{\partial \mathbf{h}_{t'}}{\partial \mathbf{h}_{t-\tau}} \tag{4.16}$$

For simple RNNs, by examining Equation 4.6, we can immediately see that all partials $\frac{\partial \mathbf{h}_{t'}}{\partial \mathbf{h}_{t-\tau}}$ are $\mathbf{0}$ except for the one satisfying $t' = t - \tau + 1$. This yields

$$\frac{\partial^+ \mathbf{h}_t}{\partial \mathbf{h}_{t-\tau}} = \frac{\partial^+ \mathbf{h}_t}{\partial \mathbf{h}_{t-\tau+1}} \frac{\partial \mathbf{h}_{t-\tau+1}}{\partial \mathbf{h}_{t-\tau}} \tag{4.17}$$

Now, by applying Equation 4.12 again to $\frac{\partial^+ \mathbf{h}_t}{\partial \mathbf{h}_{t-\tau+1}}$, and then to $\frac{\partial^+ \mathbf{h}_t}{\partial \mathbf{h}_{t-\tau+2}}$, and so on, we

trace out a path from $t-\tau$ to t, as shown in Figure 4.1, finally resulting the single term

$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} \cdots \frac{\partial \mathbf{h}_{t-\tau+2}}{\partial \mathbf{h}_{t-\tau+1}} \frac{\partial \mathbf{h}_{t-\tau+1}}{\partial \mathbf{h}_{t-\tau}} \tag{4.18}$$

which is associated with the *only* path from $t-\tau$ to t, with one factor for each edge that is encountered along the path.

Next we consider simple NARX RNNs, again by expanding Equation 4.15. From Equation 4.10, we can see that up to n_d partials are now nonzero, and that any particular partial $\frac{\partial \mathbf{h}_{t'}}{\partial \mathbf{h}_{t-\tau}}$ is nonzero if and only if $t' > t-\tau$ and t' and $t-\tau$ share an edge. Collecting these t' as the set $V_{t-\tau} = \{t' : t' > t-\tau \text{ and } (t-\tau, t') \in E\}$, we can write

$$\frac{\partial^+ \mathbf{h}_t}{\partial \mathbf{h}_{t-\tau}} = \sum_{t' \in V_{t-\tau}} \frac{\partial^+ \mathbf{h}_t}{\partial \mathbf{h}_{t'}} \frac{\partial \mathbf{h}_{t'}}{\partial \mathbf{h}_{t-\tau}} \tag{4.19}$$

We can then apply this exact same process to each $\frac{\partial^+ \mathbf{h}_t}{\partial \mathbf{h}_{t'}}$; by defining $V_{t'} = \{t'' : t'' > t' \text{ and } (t', t'') \in E\}$ for all t', we can write

$$\frac{\partial^+ \mathbf{h}_t}{\partial \mathbf{h}_{t-\tau}} = \sum_{t' \in V_{t-\tau}} \sum_{t'' \in V_{t'}} \frac{\partial^+ \mathbf{h}_t}{\partial \mathbf{h}_{t''}} \frac{\partial \mathbf{h}_{t''}}{\partial \mathbf{h}_{t'}} \frac{\partial \mathbf{h}_{t'}}{\partial \mathbf{h}_{t-\tau}} \tag{4.20}$$

By continuing this process until only partials remain, we obtain a summation over all possible paths from $t-\tau$ to t. *Each term* in the sum is a product over factors, one per edge:

$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t'''\cdots}} \cdots \frac{\partial \mathbf{h}_{t''}}{\partial \mathbf{h}_{t'}} \frac{\partial \mathbf{h}_{t'}}{\partial \mathbf{h}_{t-\tau}} \tag{4.21}$$

This analysis also applies to general NARX RNNs: the only difference is the specific sets of edges that are considered.

We therefore obtain a sum over gradient components, with each component corresponding to exactly one path from $t-\tau$ to t and being a product over its path's edges. The spectral norm corresponding to any particular path $(t - \tau \to t' \to t'' \to \cdots \to t)$ can then bounded as

$$\left\| \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t'''\cdots}} \cdots \frac{\partial \mathbf{h}_{t'}}{\partial \mathbf{h}_{t-\tau}} \right\| \leq \left\| \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t'''\cdots}} \right\| \cdots \left\| \frac{\partial \mathbf{h}_{t'}}{\partial \mathbf{h}_{t-\tau}} \right\| \leq \lambda^{n_e} \tag{4.22}$$

where λ is the maximum spectral norm of any factor and n_e is the number of edges on the path. Terms with $\lambda < 1$ diminish exponentially fast, and when all $\lambda < 1$, shortest paths dominate. It is also possible for gradient contributions to explode exponentially fast, however this problem can be remedied in practice with gradient clipping. None of the architectures discussed in this work, including LSTM, address the exploding gradient problem.

4.4 Mixed History Recurrent Neural Networks

Viewing gradient components as paths, with each component being a product with one factor per edge along the path, gives us useful insight into various RNN

architectures. When relating a loss at time t to events at time $t - \tau$, simple RNNs and LSTM contain shortest paths of length τ, while simple NARX RNNs contain shortest paths of length τ/n_d, where n_d is the number of delays.

One can envision many NARX RNN architectures with non-contiguous delays that reduce these shortest paths further. In this section we introduce one such architecture using base-2 exponential delays. In this case, for all $\tau \leq 2^{n_d-1}$, shortest paths exist with only $\log_2 \tau$ edges; and for $\tau > 2^{n_d-1}$, shortest paths exist with only $\tau/2^{n_d-1}$ edges (see Figure 4.1). Finally we must avoid the parameter and computation growth of simple NARX RNNs. We achieve this by sharing weights over delays, instead using an attention-like mechanism [62] over delays and a reset mechanism from gated recurrent units [33].

The proposed architecture, which we call mixed history RNNs (MIST RNNs), is described by

$$\mathbf{a}_t = \text{softmax}(\mathbf{W}_{ah}\mathbf{h}_{t-1} + \mathbf{W}_{ax}\mathbf{x}_t + \mathbf{b}_a) \tag{4.23}$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_{rh}\mathbf{h}_{t-1} + \mathbf{W}_{rx}\mathbf{x}_t + \mathbf{b}_r) \tag{4.24}$$

$$\mathbf{h}_t = \tanh\left(\mathbf{W}_h\left[\mathbf{r}_t \odot \sum_{i=0}^{n_d-1} a_{ti}\mathbf{h}_{t-2^i}\right] + \mathbf{W}_x\mathbf{x}_t + \mathbf{b}\right) \tag{4.25}$$

Here, $\mathbf{a}_t$ is a learned vector of n_d convex-combination coefficients and $\mathbf{r}_t$ is a reset gate. At each time step, a convex combination of delayed states is formed according to $\mathbf{a}_t$; units of this combination are reset according to $\mathbf{r}_t$; and finally the typical linear

layer and nonlinearity are applied.

4.5 Experiments: MIST RNNs for Learning Long-Term Dependencies

We compare MIST RNNs to simple RNNs, LSTM, and Clockwork RNNs. We begin with the sequential permuted MNIST task and the copy problem, synthetic tasks that were introduced to explicitly test RNNs for their ability to learn long-term dependencies [32,47,54,56,59,63]. Next we move on to 2 tasks for which it is plausible that very long-term dependencies play a role: recognizing phonemes from speech and classifying activities from smartphone motion data. We note that for all architectures involved, many variations can be applied (variational dropout, layer normalization, zoneout, etc.). We keep experiments manageable by comparing architectures without such variations.

4.5.1 Experimental Setup

All weight matrices are initialized using a normal distribution with a mean of 0 and a standard deviation of $1/\sqrt{n_h}$, where n_h is the number of hidden units. All initial hidden states (for $t < 1$) are initialized to $\mathbf{0}$. For optimization, gradients are computed using full backpropagation through time, and we use stochastic gradient

descent with a momentum of 0.9, with gradient clipping as described by [31] at 1, and with a minibatch size of 100. Biases are generally initialized to 0, but we follow best practice for LSTM by initializing the forget-gate bias to 1 [45, 64]. For Clockwork RNNs, 8 exponential periods are used, as in the original paper. For MIST RNNs, 8 delays are used. We avoid manual learning-rate tuning in its entirety. Instead we run 50 trials for each experimental configuration. In each trial, the learning rate is drawn uniformly at random in log space between 10^{-4} and 10^{1}, and initial weight matrices are also redrawn at random. We report results over the top 10% of trials according to validation-set error. (An alternative option is to report results over *all* trials. However, because the majority of trials yields bad performance for all methods, this simply blurs comparisons. See for example Figure 3 of [34], which compares these two options.)

4.5.2 Permuted MNIST

The sequential MNIST task [63] consists of classifying 28x28 MNIST images [65] as one of 10 digits, by scanning pixel by pixel – left to right, top to bottom – and emitting a label upon completion. Sequential pMNIST [63] is a challenging variant where a random permutation of pixels is chosen and applied to all images before classification. LSTM with 100 hidden units is used as a baseline, with hidden unit counts for other architectures chosen to match the number of parameters. Means and standard deviations are computed using the top 5 randomized trials out of 50

(ranked according to performance on the validation set), with random learning rates and initializations.

Test error rates are shown in Table 4.1. Here, MIST RNNs outperform simple RNNs, LSTM, and Clockwork RNNs by a large margin. We remark that our LSTM error rates are consistent with best previously-reported values, such as the error rates of 9.8% in [66] and 12% in [47], which also use 100 hidden units. One may also wonder if the difference in performance is due to hidden-unit counts. To test this we also increased the LSTM hidden unit count to 139 (to match MIST RNNs), and continued to increase the capacity of each model further. MIST RNNs significantly outperform LSTM in all cases.

We also used this task to visualize gradient magnitudes as a function of τ (the distance from the loss which occurs at time $t = 784$). Gradient norms for all methods were averaged over a batch of 100 random examples early in training; see Figure 4.2. Here we can see that simple RNNs and LSTM capture essentially no learning signal from steps that are far from the loss. To validate this claim further, we repeated the 512-unit LSTM and MIST RNN experiments, but using only the last 200 permuted pixels (rather than all 784). LSTM performance remains the same (7.4% error, within 1 standard deviation) whereas MIST RNN performance drops by 15 standard deviations (6.0% error).

Table 4.1: **Test-set error rates for sequential pMNIST classification.** Hidden unit counts (n_h) vary to match parameter counts with LSTM (approx. 42,000 parameters), except models marked with + (which have more parameters). α^* denotes the optimal learning rate according to validation error.

	n_h	$\log_{10} \alpha^*$	Error Rate (%)
Simple RNNs	198	-2.27 ± 0.10	12.9 ± 0.8
LSTM	100	-1.11 ± 0.11	10.4 ± 0.7
Clockwork RNNs	256	-1.91 ± 0.23	15.7 ± 1.2
MIST RNNs	139	-1.35 ± 0.08	5.5 ± 0.2
LSTM$^+$	139	-0.90 ± 0.26	8.8 ± 0.6
LSTM$^+$	512	-1.08 ± 0.18	7.6 ± 0.7
MIST RNNs$^+$	512	-1.19 ± 0.13	4.5 ± 0.1

4.5.3 The Copy Problem

The copy problem is a synthetic task that explicitly challenges a network to store and reproduce information from the past. Our setup follows [47], which is in turn based on [32]. An input sequence begins with L relevant symbols to be copied, is followed by a delay of $D - 1$ special blank symbols and 1 special go symbol, and ends with L additional blank symbols. The corresponding target sequence begins with $L + D$ blank symbols and ends with a copy of the relevant symbols from the inputs (in the same order). We run experiments with copy delays of $D = 50$, 100, 200, and 400. LSTM with 100 hidden units is used as a baseline, with hidden unit counts for other architectures chosen to match the number of parameters. Additional experimental details can be found in the appendix.

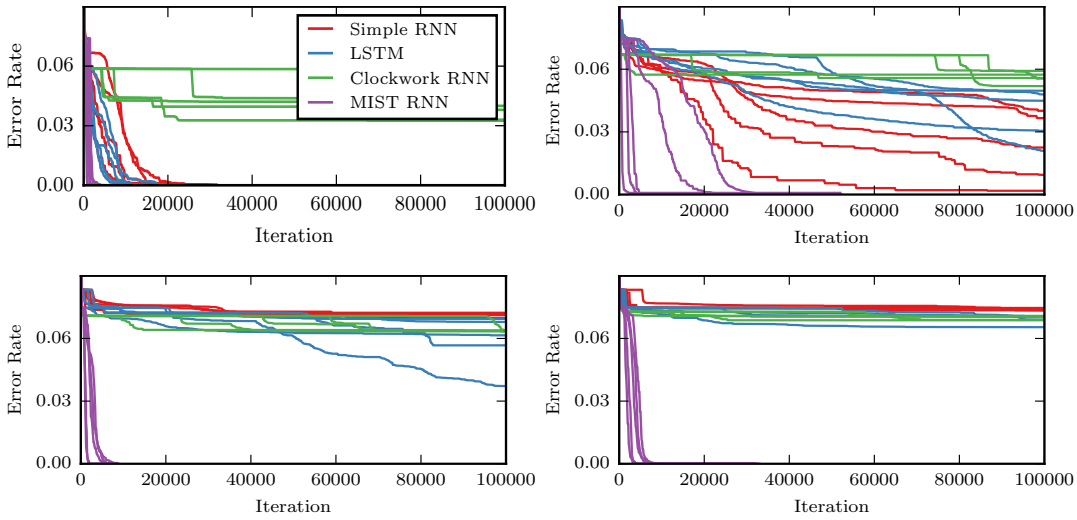


Figure 4.3: **Validation curves for the copy problem with copy delays of 50 (upper left), 100 (upper right), 200 (lower left), and 400 (lower right).** MIST RNNs, unlike simple RNNs, LSTM, and Clockwork RNNs, are able to learn to solve the copy problem even for long delays.

Results are shown in Figure 4.3, showing validation curves of the top 5 randomized trials out of 50, with random learning rates and initializations. With a short copy delay of $D = 50$, we can see that all methods other than Clockwork RNNs can solve the task in a reasonable amount of time. However, as the copy delay D is increased, we can see that simple RNNs and LSTM become unable to learn a solution, whereas MIST RNNs are relatively unaffected. We also note that our LSTM results are consistent with those in [47, 59].

Note that Clockwork RNNs are expected to fail for large delays (for example, the second symbol can only be seen by the highest-frequency partition, so learning to copy this symbol will fail for precisely the same reason that simple RNNs fail). However, here they also fail for short delays, which is surprising because the high-

Table 4.2: **Test-set error rates for TIMIT phoneme recognition.** Hidden unit counts (n_h) vary to match parameter counts with LSTM (approx. 44,000 parameters). α^* denotes the optimal learning rate according to validation error.

	n_h	$\log_{10} \alpha^*$	Error Rate (%)
Simple RNNs	197	-1.09 ± 0.25	34.1 ± 0.3
LSTM	100	-0.63 ± 0.06	32.1 ± 0.2
Clockwork RNNs	248	-1.03 ± 0.31	38.2 ± 0.4
MIST RNNs	139	-0.91 ± 0.16	32.0 ± 0.3

speed partition resembles a simple RNN. We hypothesized that this failure is due to hidden unit counts / parameter counts: here, the high-frequency partition is allocated only 256 / 8 = 32 hidden units. To test this hypothesis, we reran the Clockwork RNN experiments with 1024 hidden units, so that 128 are allocated to the high-frequency partition. Indeed, under this configuration (with 10x as many parameters), Clockwork RNNs do solve the task for a delay of $D = 50$ and fail to solve the task for all higher delays, thus behaving like simple RNNs.

4.5.4 Phoneme Recognition

Here we consider the task of online framewise phoneme recognition using the TIMIT corpus [67]. Each frame is originally labeled as 1 of 61 phonemes. We follow common practice and collapse these into a smaller set of 39 phonemes [68], and we include glottal stops to yield 40 classes in total. We follow [34] for data preprocessing and [69] for training, validation, and test splits. LSTM with 100 hidden units is used

as a baseline, with hidden unit counts for other architectures chosen to match the number of parameters. Means and standard deviations are computed using the top 5 randomized trials out of 50 (ranked according to performance on the validation set), with random learning rates and initializations. Other experimental details can be found in the appendix. Table 4.2 shows that LSTM and MIST RNNs perform nearly identically, which both outperform simple RNNs and Clockwork RNNs.

4.5.5 Activity Recognition from Smartphones

Here we consider the task of sequence classification from smartphones using the MobiAct (v2.0) dataset [70]. The goal is to classify each sequence as jogging, running, sitting down, etc., using smartphone motion data over time. Approximately 3,200 sequences were collected from 67 different subjects. We use the first 47 subjects for training, the next 10 for validation, and the final 10 for testing. Means and standard deviations are computed using the top 5 randomized trials out of 50 (ranked according to performance on the validation set), with random learning rates and initializations. Other experimental details can be found in the appendix. Results are shown in Table 4.3. Here, MIST RNNs outperform all other methods, including LSTM and LSTM$^+$, a variant with the same number of hidden units and twice as many parameters.

Table 4.3: **Test-set error rates for MobiAct activity classification.** Hidden unit counts (n_h) vary to match parameter counts with LSTM (approx. 44,000 parameters), with the exception of LSTM$^+$ (approx. 88,000 parameters). α^* denotes the optimal learning rate according to validation error.

	n_h	$\log_{10} \alpha^*$	Error Rate (%)
Simple RNNs	203	-1.91 ± 0.18	49.2 ± 2.7
LSTM	100	-0.89 ± 0.12	38.8 ± 1.5
LSTM$^+$	141	-0.45 ± 0.17	37.8 ± 2.1
Clockwork RNNs	256	-1.09 ± 0.31	40.3 ± 4.8
MIST RNNs	141	-1.39 ± 0.21	29.0 ± 5.2

4.6 Experiments: MIST RNNs for Surgical Activity Recognition

Here we consider NARX RNNs for surgical activity recognition, using the same experimental design as Chapter 3, alongside simple RNNs, LSTM, and GRUs. We remark that we also considered Clockwork RNNs in preliminary experiments, but performance was limited, achieving 2-3 times higher error rates (22.5% at best) than LSTM and GRUs in all experiments, and so Clockwork RNNs were not considered in later experiments. For clarity, we briefly recap the datasets and experimental details below. In a larger context, the primary goal of this section is to determine whether or not the ability to learn extremely long-term dependencies, by using MIST RNNs, can improve surgical-activity-recognition performance.

For maneuver recognition, we use the Minimally Invasive Surgical Training and

Innovation Center – Science of Learning (MISTIC-SL) dataset [41], and for gesture recognition, we use the JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS) dataset [19, 42]. Both datasets contain kinematic data collected from a *da Vinci* surgical system, along with activity annotations over time, provided manually by experts. In both cases, we use 14 kinematic signals as input: velocities, angular velocities, and gripper angle, all for both the left and right hands. Figure 3.2 shows frames over time that are representative of both MISTIC-SL and JIGSAWS (taken from MISTIC-SL). The performance metrics used are frame-wise error rate (the percentage of incorrectly-labeled frames) and segment-level edit distance (or Levenshtein distance, the number of operations required to transform a segment-level prediction sequence into a segment-level ground-truth sequence). For hyperparameter analysis, we rely on the functional ANOVA (fANOVA) framework [43]. We focus on 4 key hyperparameters: the number of hidden units per layer, n_h, the number of layers, n_l, the shared dropout probability applied to the input kinematics and to the hidden states of the final layer, d, and the learning rate, α. For each RNN architecture, hyperparameters are explored using random search [44] with 200 trials. For each trial, $\log_2 n_h$ is chosen uniformly from $\{4, 5, \ldots, 9\}$; n_l is chosen uniformly from $\{1, 2, 3\}$; d is chosen uniformly from $[0, 0.5]$; and $\log_{10} \alpha$ is chosen uniformly from $[-4, -2]$. And, as in all experiments in this thesis, focus is on generalizing to new subjects, using a leave-one-user-out (LOUO) evaluation setup. More detail is included in Chapter 3.

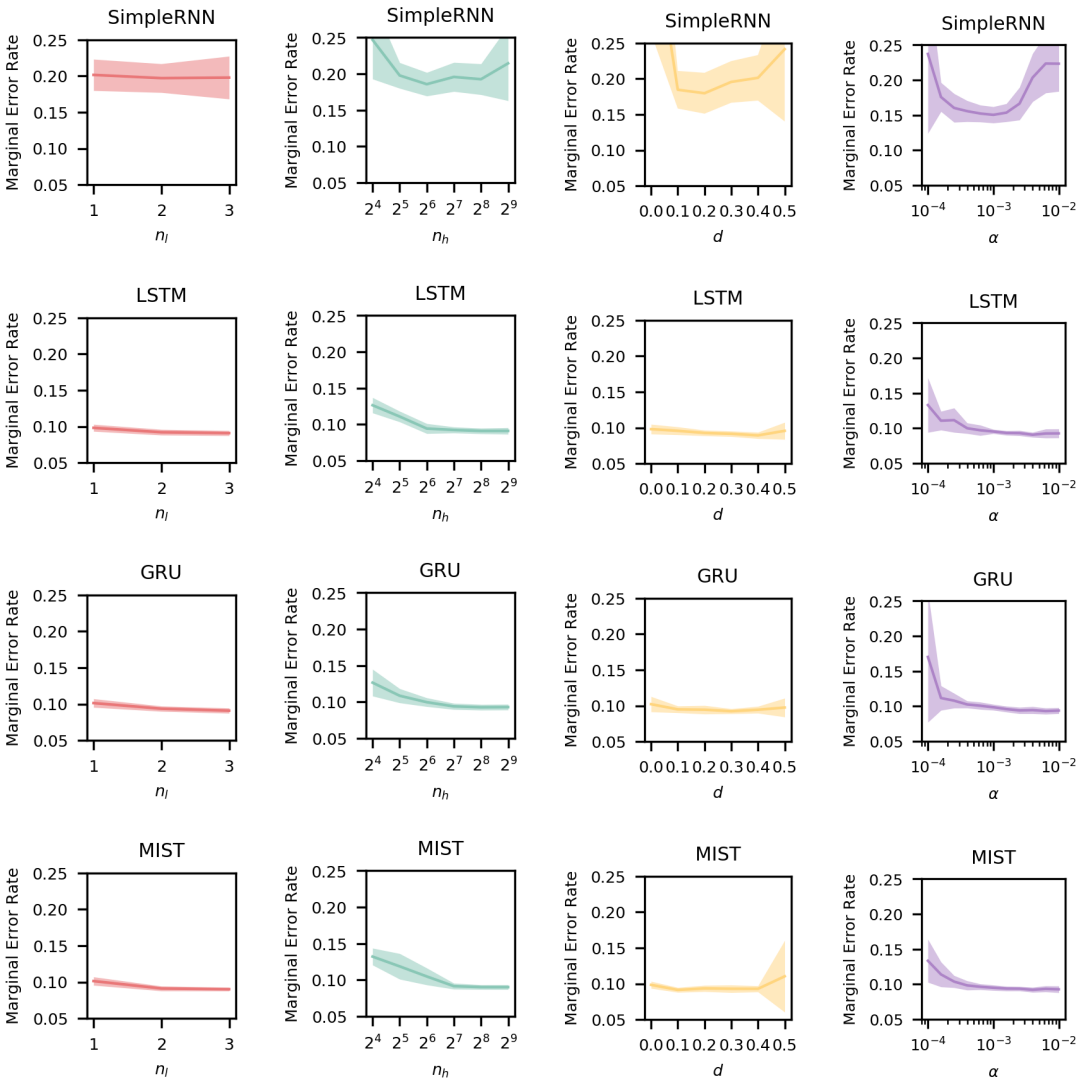


Figure 4.4: **Estimated marginal error rate for each hyperparameter for MIST RNNs, shown alongside the other RNN architectures from Chapter 3.** The dark line represents the mean and the top and bottom edges represent one standard deviation. From left to right, the hyperparameters are the number of layers n_l, the number of hidden units n_h, the dropout probability d, and the learning rate α.

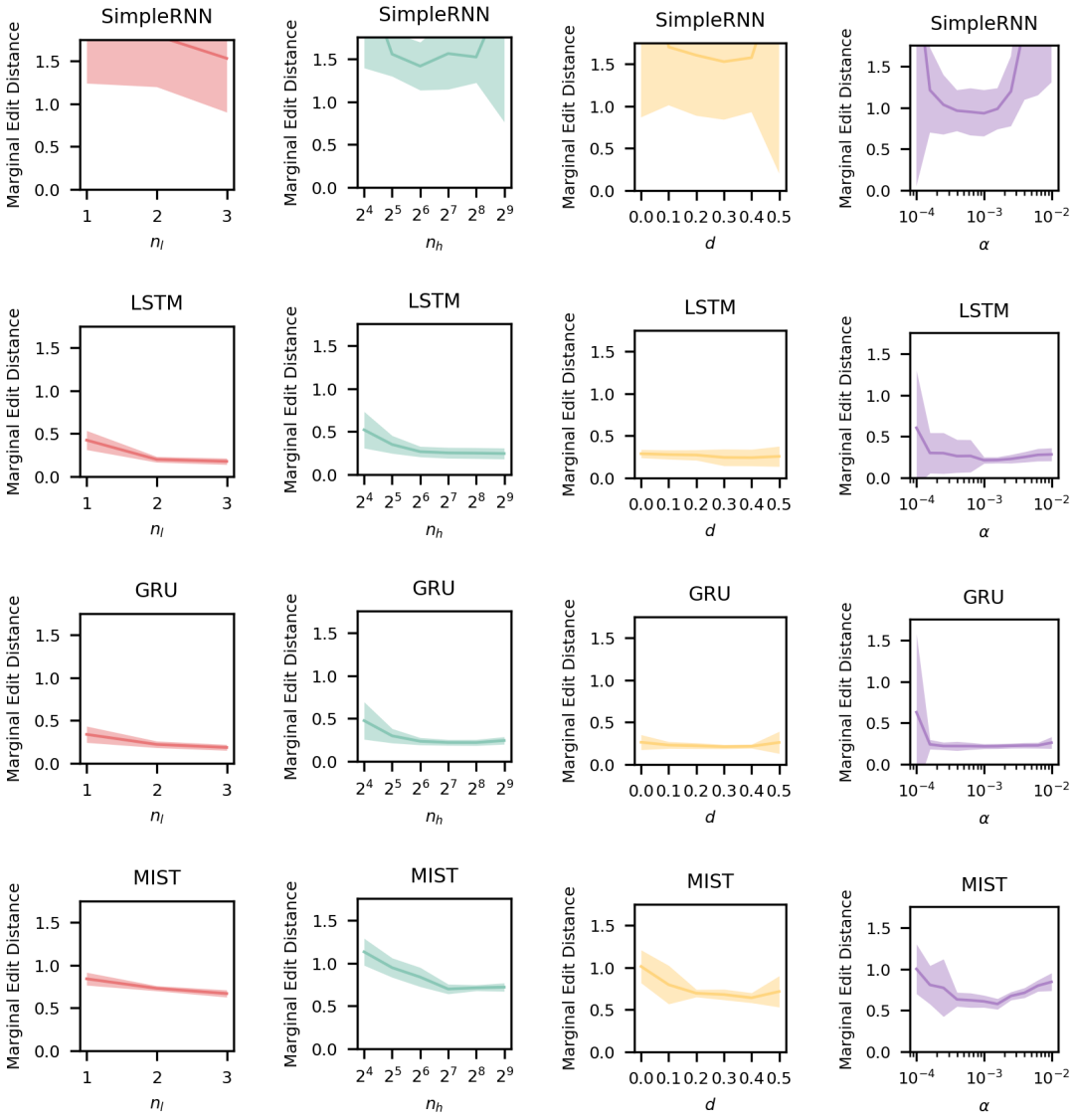


Figure 4.5: **Estimated marginal edit distance for each hyperparameter for MIST RNNs, shown alongside the other RNN architectures from Chapter 3.** The dark line represents the mean and the top and bottom edges represent one standard deviation. From left to right, the hyperparameters are the number of layers n_l, the number of hidden units n_h, the dropout probability d, and the learning rate α.

Validation performance, alongside previous results for simple RNNs, LSTM, and GRUs, is summarized in Figures 4.4, 4.5, and 4.6. The final hyperparameters chosen

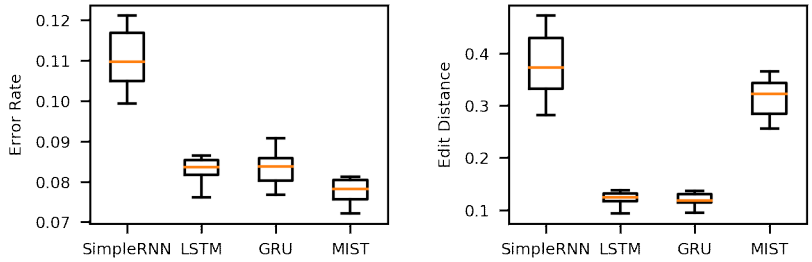


Figure 4.6: **Box plots summarizing error rates (left) and normalized edit distances (right) for MIST RNNs, shown alongside the other RNN architectures from Chapter 3, computed from the top 10% of validation runs.** Each box represents the 25th, 50th, and 75th percentiles, while the whiskers represent minimum and maximum values.

using the validation set for MIST RNNs, as illustrated in Table 3.1 for simple RNNs, LSTM, and GRUs, are $n_l = 2$ layers, $n_h = 256$ hidden units, a dropout probability of $d = 0.4$, and a learning rate of $\alpha = 10^{-2.8}$. This configuration has approximately 260k parameters and yielded a validation error rate of 7.0% and a validation edit distance of 37.3%.

Table 4.4 shows test-set performance for maneuver recognition, and Table 4.5 for gesture recognition. MIST RNNs result in error rates that are slightly higher for maneuver recognition (9.7%, vs. approximately 8.6% for LSTM and GRUs), and that are essentially identical for gesture recognition (15.3%). However, with regard to edit distance, MIST RNNs perform 2-3x worse. This is an interesting result, and one that we attribute to the implicit biases present in LSTM and GRUs, but not in simple RNNs or MIST RNNs. For the purposes of discussion, consider LSTM, and in particular Equation 4.7. Notice that the new memory cell is a simple combination of the old cell along with a new update, *without* any mandatory linear

transformation followed by a mandatory nonlinearity. This is especially relevant since at the beginning of training the forget gate is purposely initialized to have elements that are close to 1.0, in order to mitigate the vanishing gradient problem. In contrast, MIST RNNs (and simple RNNs) form the new hidden state through a mandatory linear transformation and mandatory nonlinearity (Equation 4.25), and so there is no bias toward hidden representations that are smooth over time, and in turn no bias toward activity predictions that are smooth over time. From a more global perspective, however, this section is primarily concerned with the impact of learning long-term dependencies on activity-recognition performance, as measured by error rate, since during training, we are optimizing a relaxation of error rate, and not edit distance. Given that MIST RNNs are capable of outperforming LSTM for problems requiring extremely long-term dependencies (as shown in the sections above), a likely conclusion is that gesture recognition and maneuver recognition are both dominated by short- to medium-length dynamics. In our experiments, LSTM and GRUs maintain non-negligible gradient magnitudes for up to approximately 50 time steps, which is in turn approximately 10 seconds in our experiments; and the longer reach of MIST RNNs, which maintain non-negligible gradient magnitudes for hundreds of time steps (see Figure 4.2), is not beneficial.

Table 4.4: **MISTIC-SL mean error rates and normalized edit distances for MIST RNNs, alongside results for simple RNNs, LSTM, and GRUs.** All results are averaged over users in a leave-one-user-out evaluation setup.

	Error Rate (%)	Norm. Edit Dist. (%)	Norm. Edit Dist.* (%)
Bidir SimpleRNN	11.6	31.7	26.9
Bidir LSTM	8.7	12.1	12.7
Bidir GRU	8.6	9.3	10.0
Bidir MIST	9.7	33.2	27.0

Table 4.5: **JIGSAWS mean error rates and normalized edit distances for MIST RNNs, alongside results for simple RNNs, LSTM, and GRUs.** All results are averaged over users in a leave-one-user-out evaluation setup.

	Error Rate (%)	Norm. Edit Dist. (%)	Norm. Edit Dist.* (%)
Bidir SimpleRNN	17.9	17.3	21.1
Bidir LSTM	15.3	8.4	11.9
Bidir GRU	15.2	8.4	11.5
Bidir MIST	15.3	25.7	23.6

Qualitative results are also included, shown for maneuver recognition in Figure 4.7 and for gesture recognition in Figure 4.8. These primarily highlight the superior regularity of LSTM and GRUs. These results are all displayed using a single trial that is as representative as possible. Additional results, showing trials with median error rate and median edit distance for each architectures, are included in the appendix.

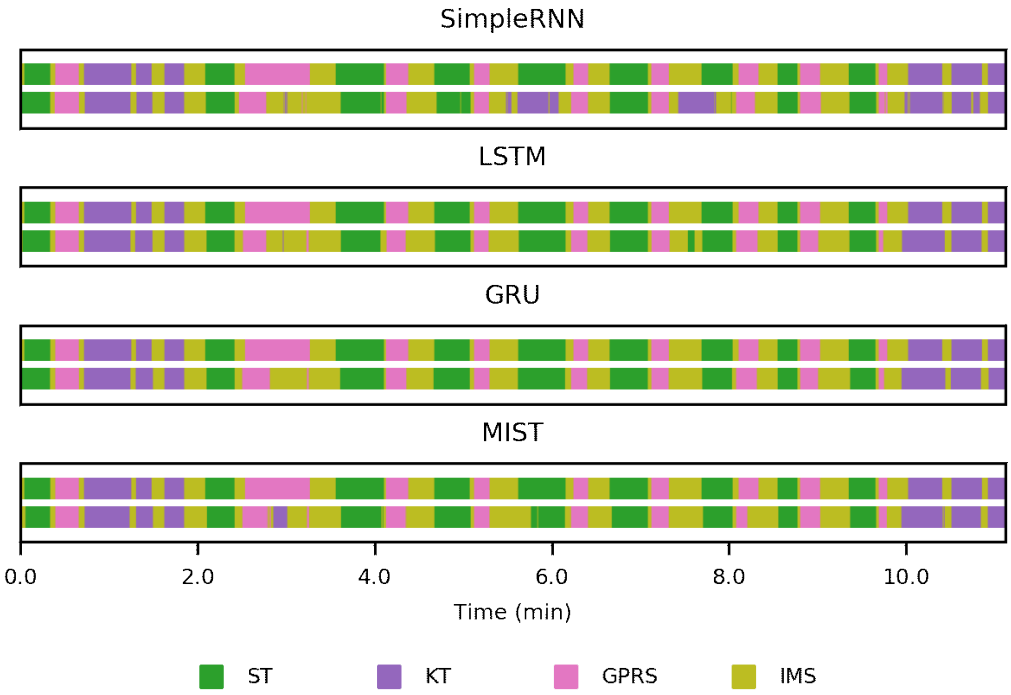


Figure 4.7: **Qualitative MISTIC-SL Results (Maneuver Recognition) for MIST RNNs, alongside simple RNNs, LSTM, and GRUs.** Ground truth labels are shown directly above predicted labels for each architecture. A single trial that is as representative as possible is shown to ease comparison; see the appendix for more results.

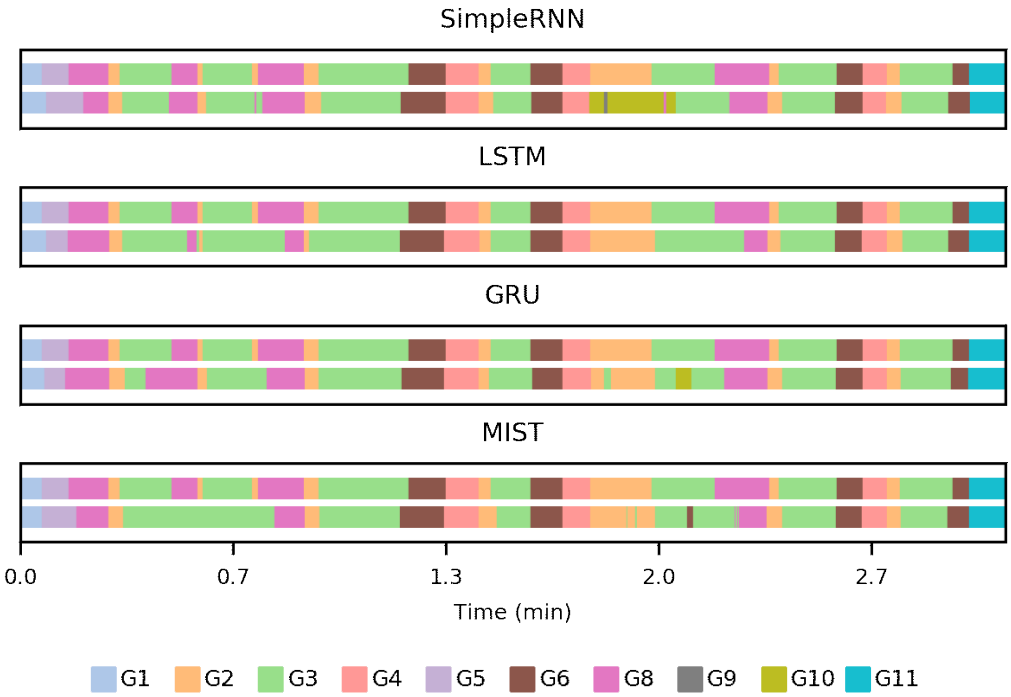


Figure 4.8: **Qualitative JIGSAWS Results (Gesture Recognition) for MIST RNNs, alongside simple RNNs, LSTM, and GRUs.** Ground truth labels are shown directly above predicted labels for each architecture. A single trial that is as representative as possible is shown to ease comparison; see the appendix for more results.

4.7 Conclusions

In this chapter, we introduced mixed history RNNs, a recurrent neural network architecture that is capable of learning dependencies across extremely large time scales. We motivated this architecture through a theoretical analysis and empirically demonstrated that they outperform the most widely-used architectures to date when such long-term dependencies are required.

In the context of surgical activity recognition, however, we found that the ability to learn such long-term dependencies does not improve performance. In other words, the window afforded by LSTM and GRUs is sufficient; and further, the implicit bias of LSTM and GRUs toward smooth predictions is beneficial for surgical activity recognition, in which activities span seconds (in the case of gestures) or 10s of seconds (in the case of maneuvers). Looking forward, we will focus primarily on LSTM, and we will shift our attention to a different aspect of surgical activity recognition: learning meaningful reprsentations without annotations, and leveraging these learned representations for activity recognition when few annotations are available.

4.8 Appendix

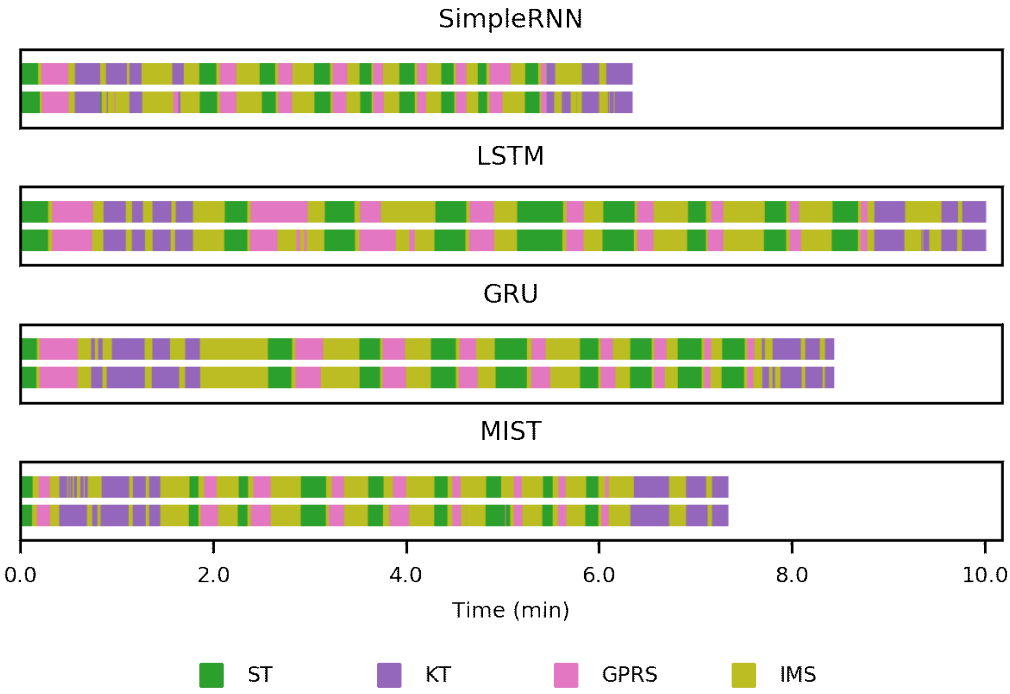


Figure 4.9: **Qualitative results: MISTIC-SL trials with median error rate for MIST RNNs, alongside simple RNNs, LSTM, and GRUs.** Ground truth labels are shown above predicted labels. From top to bottom, the error rates are 11.3%, 7.7%, 7.0%, and 8.6% (with normalized edit distances of 30.5%, 20.3%, 6.8%, and 20.3%).

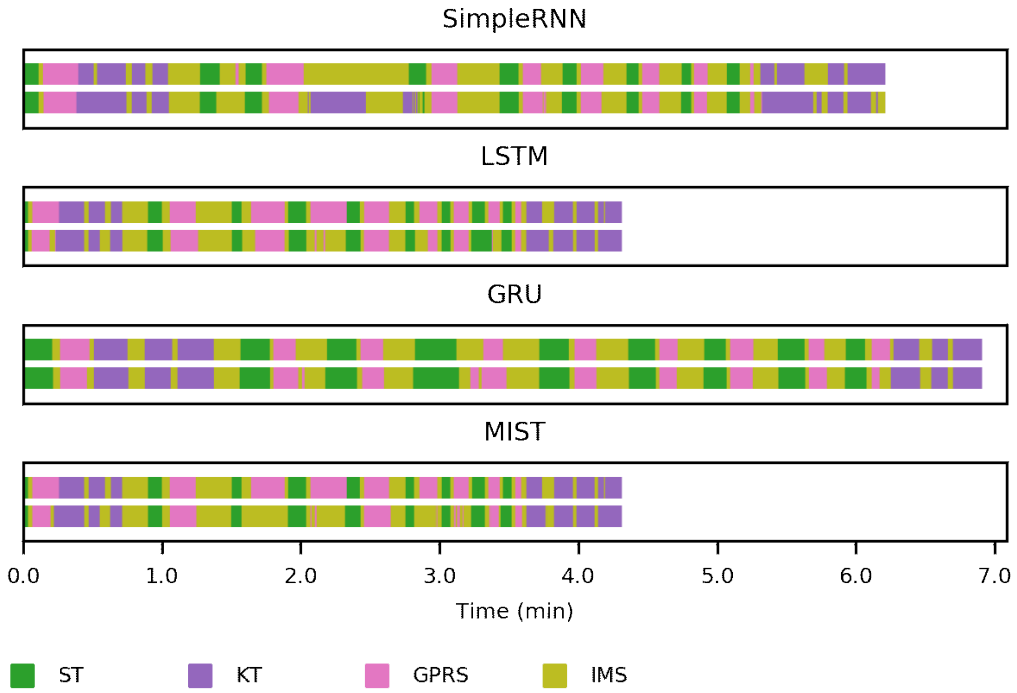


Figure 4.10: **Qualitative results: MISTIC-SL trials with median edit distance for MIST RNNs, alongside simple RNNs, LSTM, and GRUs.** Ground truth labels are shown above predicted labels. From top to bottom, the normalized edit distances are 30.5%, 10.2%, 6.8%, and 28.8% (with error rates of 15.8%, 16.3%, 6.4%, and 21.1%).

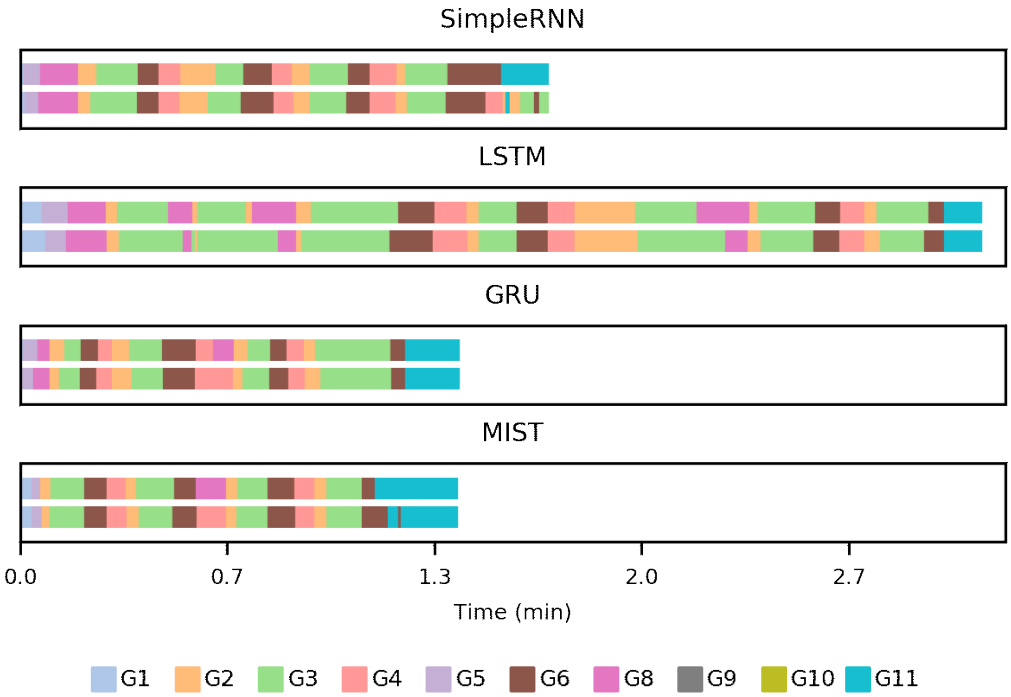


Figure 4.11: **Qualitative results: JIGSAWS trials with median error rate for MIST RNNs, alongside simple RNNs, LSTM, and GRUs.** Ground truth labels are shown above predicted labels. From top to bottom, the error rates are 16.1%, 12.9%, 12.3%, and 12.9% (with normalized edit distances of 16.2%, 5.4%, 2.7%, and 8.1%).

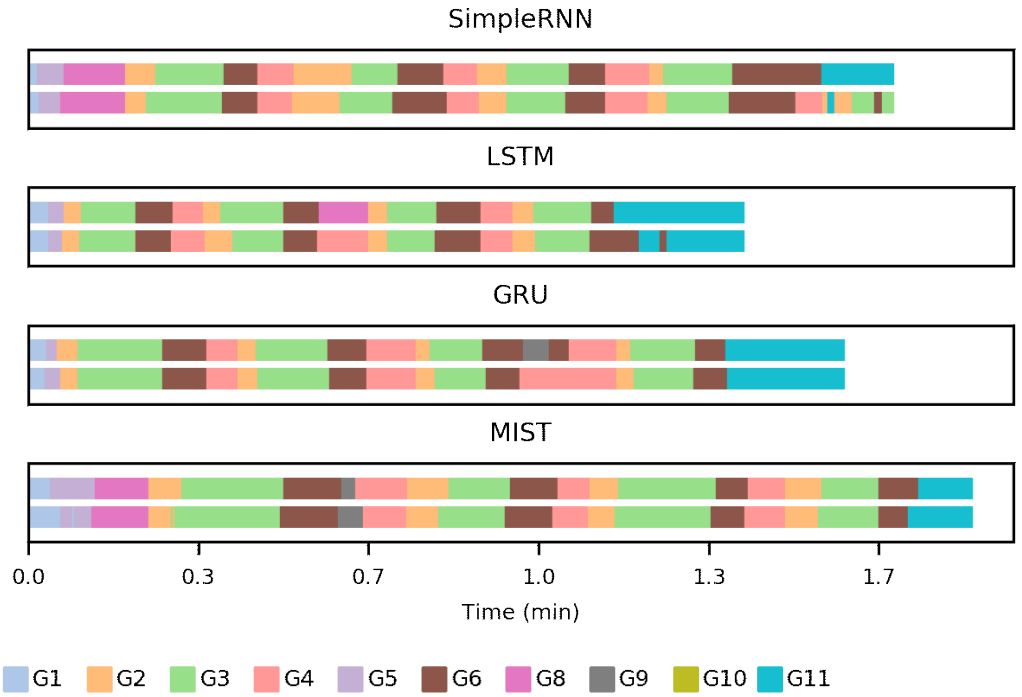


Figure 4.12: **Qualitative results: JIGSAWS trials with median edit distance for MIST RNNs, alongside simple RNNs, LSTM, and GRUs.** Ground truth labels are shown above predicted labels. From top to bottom, the normalized edit distances are 16.2%, 8.1%, 5.4%, and 10.8% (with error rates of 16.1%, 15.0%, 9.0%, and 9.4%).

Chapter 5

Future Prediction for Data-Efficient Surgical Activity Recognition

In this chapter, we deviate from the common assumption that an abundance of annotated data is available for training. First, we consider representation learning in a completely unsupervised fashion, and we demonstrate that future prediction is a promising auxilliary task for this purpose. Here, we show that this can even lead to the *discovery* of surgical activities from motion data; and that the representations that results yield state-of-the-art performance for motion-based search. Next, we show that leveraging these learned representations in the downstream task of activity recognition improves performance when annotations are scarce. And last, we show

that these methods generalize well to other tasks, considering both simple, isolated experiments involving pendulums and the more difficult task of recognizing phonemes from speech data.

5.1 Introduction

In Chapter 3, we demonstrated that it is feasible to recognize high-level surgical activities that are consistent with current training curricula (*maneuvers*, e.g. knot tying and suturing). However, following essentially all prior work in surgical activity recognition, we assumed that an abundance of manually annotated sequences are available for training. Unlike the surgical-motion data itself, these annotations are difficult to acquire, are often subjective, and may be of variable quality. In addition, many questions surrounding annotations remain open. For example, should they be collected at the low level of gestures [19], at the higher level of maneuvers [42], or at some other granularity? Do annotations transfer between surgical tasks? And how consistent are annotations among experts?

In this chapter, we focus on learning meaningful and useful representations of surgical motion from the data itself, *without annotations*, and on leveraging these learned representations for data-efficient activity recognition, *when few annotations are available.*

An important question is what model and/or tasks we should consider for learn-

ing meaningful representations. A predominant theory in neuroscience is that many mammals, including humans, are constantly predicting its incoming signals – and that these predictions are even an integral part of perception itself [71–75]. This theory, known as *predictive coding*, asserts that higher-level (more abstract) layers of the brain are constantly trying to predict the inputs to the more concrete layers below; and that the lower levels do not passively transmit its inputs to higher levels, but rather transmit *error signals* – the portions of its inputs that the higher levels failed to predict. In this chapter, rather than mimicking our very limited understanding of how this functionality takes place in the brain, we mimic the simple idea of predicting future observations, and we do so within frameworks that are based on recurrent neural networks.

We begin by introducing a window-based future-prediction model which predicts a window of future motion from a window of previous motion. From a qualitative perspective, we show that this model is able to *discover* high-level maneuvers, in the sense that low-dimensional visualizations of the representations show that clusters tend to form according to high-level activity. And from a quantitative perspective, we show that these representations, in combination with a simple similarity metric (cosine similarity), achieve state-of-the-art performance for querying a database of surgical motion with motion-based queries. Importantly, future-prediction-based representations outperform popular autoencoder-based approaches, even when used within a more-complex pipeline for the same application, as in [76].

Next, we introduce an RNN-based generative model for future prediction. This model relaxes simplifying assumptions made in the previous window-based model (namely probabilistic independence over time), and additionally does not require operating at the window level, which requires window size to be specified as a hyperparameter. Here, we focus on the predominant task of interest: surgical activity recognition. We find that recognition is possible *even when only one annotated sequence is available*, and that leveraging learned representations, prior to the recognition phase, leads to large boosts in recognition performance, especially in the annotation-scarce regime.

5.2 Prior Work

Little prior work exists for unsupervised learning for surgical motion. The most relevant prior work is [76], which encodes short windows of kinematics using denoising autoencoders, and which uses these representations along with a custom dynamic-time-warping technique to search a database using motion-based queries. This is the primary baseline in Section 5.4. Other unsupervised approaches include activity alignment under the assumption of identical structure [42] and activity segmentation using hand-crafted pipelines [77], structured probablistic models [78], and clustering [79]. In the context of data-efficient activity recognition, even less prior work exists. To our knowledge, the only exceptions have been in the form of preprints, and have

focused on video-based recognition rather than motion-based recognition [80–82].

5.3 A Window-Based Future-Prediction Model

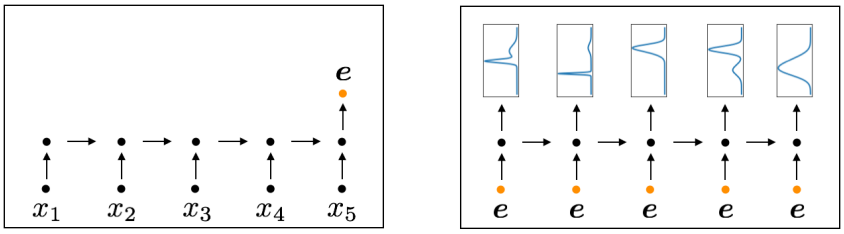
Here we introduce a model for predicting a window of future motion from a window of past motion. More precisely, letting $\mathbf{X}_p \equiv \{\mathbf{x}_t\}_1^{T_p}$ denote a subsequence of kinematics from the past and $\mathbf{X}_f \equiv \{\mathbf{x}_t\}_{T_p+1}^{T_p+T_f}$ denote the kinematics that follow, we model the conditional distribution $p(\mathbf{X}_f \mid \mathbf{X}_p)$. This is accomplished with an architecture that combines an RNN encoder-decoder with mixture density networks, as illustrated in Figure 5.1. We have discussed recurrent neural networks at length in Chapters 2 and 4. So that LSTM can easily be referenced below, we will refer to the LSTM mapping defined in Chapter 2, and in particular by Equations 2.16 – 2.21, as

$$\mathbf{h}_t = \text{LSTM}(\mathbf{x}_{t-1}, \mathbf{h}_{t-1}) \tag{5.1}$$

where the memory cell $\mathbf{c}_t$ is omitted for brevity.

For reference, we remark that RNN encoder-decoders [33] were introduced in machine translation to encode a source sentence in one language and decode it in another language, by modeling the discrete distribution $p(\text{target sentence} \mid \text{source sentence})$. We proceed similarly, by modeling the continuous conditional distribution $p(\mathbf{X}_f \mid \mathbf{X}_p)$, using LSTM for both the encoder and the decoder, as illustrated in Figure 5.1.

The encoder LSTM maps $\mathbf{X}_p$ to a series of hidden states by iteratively apply-



(a) Encoding the Past (b) Decoding the Future

Figure 5.1: **The window-based encoder-decoder architecture used for future prediction.** Here, a single signal with lengths $T_p = T_f = 5$ is shown for visualization. More accurately, each $\mathbf{x}_t \in \mathbb{R}^{n_x}$, and each time step in the future yields a multivariate mixture.

ing Equation 5.1; and the final hidden state is used as our fixed-length encoding of $\mathbf{X}_p$. Collecting the encoder's weights and biases into $\boldsymbol{\theta}^{(\text{enc})}$, we can summarize this encoding operation as

$$\mathbf{e} \equiv \mathbf{h}_{T_p}^{(\text{enc})} = f(\mathbf{X}_p; \boldsymbol{\theta}^{(\text{enc})}) \tag{5.2}$$

Similarly, the LSTM decoder, with its own parameters $\boldsymbol{\theta}^{(\text{dec})}$, maps $\mathbf{e}$ to a series of hidden states, where hidden state t is used to decode the kinematics at time step t of the future. The simplest possible estimate is then $\hat{\mathbf{x}}_t = \mathbf{W}\,\mathbf{h}_t^{(\text{dec})} + \mathbf{b}$, where training equates to minimizing sum-of-squares error. However, this approach corresponds to maximizing likelihood under a unimodal Gaussian, which is insufficient because distinct futures are blurred into one (see Figure 5.2).

In general, the vast majority of neural-network training is accomplished by minimizing a loss that corresponds to maximizing likelihood under some probabilistic model, e.g. a unimodal Gaussian in the case of mean-squared-error loss or a categorical distribution in the case of cross-entropy loss. Here, we will leverage mixture

density networks (MDNs) [83], which generalize this idea and allow neural networks to produce conditional distributions which are mixtures, or in other words which are inherently multimodal. We associate each time step of the future with its own mixture of multivariate Gaussians, with parameters that depend on $\mathbf{X}_p$ through the encoder and decoder. For each time step, every component c is associated with a mixture coefficient $\pi_t^{(c)}$, a mean $\boldsymbol{\mu}_t^{(c)}$, and a diagonal covariance matrix with entries collected in $\mathbf{v}_t^{(c)}$. These parameters are computed via

$$\boldsymbol{\pi}_t(\mathbf{h}_t^{(\text{dec})}) = \text{softmax}(\mathbf{W}_\pi \, \mathbf{h}_t^{(\text{dec})} + \mathbf{b}_\pi) \tag{5.3}$$

$$\boldsymbol{\mu}_t^{(c)}(\mathbf{h}_t^{(\text{dec})}) = \mathbf{W}_\mu^{(c)} \, \mathbf{h}_t^{(\text{dec})} + \mathbf{b}_\mu^{(c)} \tag{5.4}$$

$$\mathbf{v}_t^{(c)}(\mathbf{h}_t^{(\text{dec})}) = \text{softplus}(\mathbf{W}_v^{(c)} \, \mathbf{h}_t^{(\text{dec})} + \mathbf{b}_v^{(c)}) \tag{5.5}$$

where the softplus is used to ensure that $\mathbf{v}_t^{(c)}$ has all positive elements and where the softmax is used to ensure that $\boldsymbol{\pi}_t$ has positive elements that sum to 1.

We emphasize that all $\pi_t^{(c)}$, $\boldsymbol{\mu}_t^{(c)}$ and $\mathbf{v}_t^{(c)}$ depend implicitly on $\mathbf{X}_p$ and on the encoder's and decoder's parameters through $\mathbf{h}_t^{(\text{dec})}$, and that the individual components of $\mathbf{x}_t$ are *not* conditionally independent under this model. However, in order to capture global context rather than local properties such as smoothness, we do not condition each $\mathbf{x}_{t+1}$ on $\mathbf{x}_t$; instead, we condition each $\mathbf{x}_t$ only on $\mathbf{X}_p$ and assume

independence over time steps. Our final model is then

$$p(\mathbf{X}_f \mid \mathbf{X}_p) = \prod_{\mathbf{x}_t \in \mathbf{X}_f} \sum_c \pi_t^{(c)}(\mathbf{h}_t^{(\text{dec})}) \, \mathcal{N}\Big(\mathbf{x}_t \,;\; \boldsymbol{\mu}_t^{(c)}(\mathbf{h}_t^{(\text{dec})}), \mathbf{v}_t^{(c)}(\mathbf{h}_t^{(\text{dec})})\Big) \tag{5.6}$$

Given past, future pairs $(\mathbf{X}_p^{(n)}, \mathbf{X}_f^{(n)})$, training is carried out by minimizing the negative log likelihood $-\sum_n \log p(\mathbf{X}_f^{(n)} \mid \mathbf{X}_p^{(n)}; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is a collection of all parameters from the encoder LSTM, the decoder LSTM, and the decoder outputs. This is carried out using stochastic gradient descent. We remark that some care must be taken in order to maintain numerical stability to optimize this objective; please see the Appendix for details. We note that the encoder, the decoder, the decoder's outputs, and the negative log likelihood are all constructed within a single computation graph, and we can differentiate our loss with respect to all parameters automatically and efficiently using backpropagation through time [60]. Our implementation is based on PyTorch.

5.4 Experiments: Learning Representations Without Annotations

Here we carry out two sets of experiments. First, we compare the predictions and encodings from our future-prediction model equipped with mixture density networks, which we refer to as FP MDN, with two baseline versions: FP -MDN, which focuses

on future prediction without MDNs, and -FP MDN, which instead of predicting the future learns to compress and reconstruct the past in an autoencoder-like fashion. Second, we compare these approaches in an information-retrieval setting alongside the state-of-the-art approach [76].

5.4.1 Dataset

In this section, we focus on manuever recognition, and we recap the details of the Minimally Invasive Surgical Training and Innovation Center - Science of Learning (MISTIC-SL) dataset here. This dataset focuses on minimally-invasive, robot-assisted surgery using a *da Vinci* surgical system, in which trainees perform a structured set of tasks (see Fig. 5.4). We follow [76] and only use data from the 15 right-handed trainees in the study. Each trainee performed between 1 and 6 trials, for a total of 39 trials. We use 14 kinematic signals in all experiments: velocities, rotational velocities, and the gripper angle of the tooltip, all for both the left and right hands. In addition, experts manually annotated the trials so that all moments in time are associated with 1 of 4 high-level activities: Suture Throw (ST), Knot Tying (KT), Grasp Pull Run Suture (GPRS), or Intermaneuver Segment (IMS). We emphasize that these labels are not used in any way to obtain the encodings; they are used only for visualization, and in particular to show that future prediction is able to largely discover clusters of maneuvers.

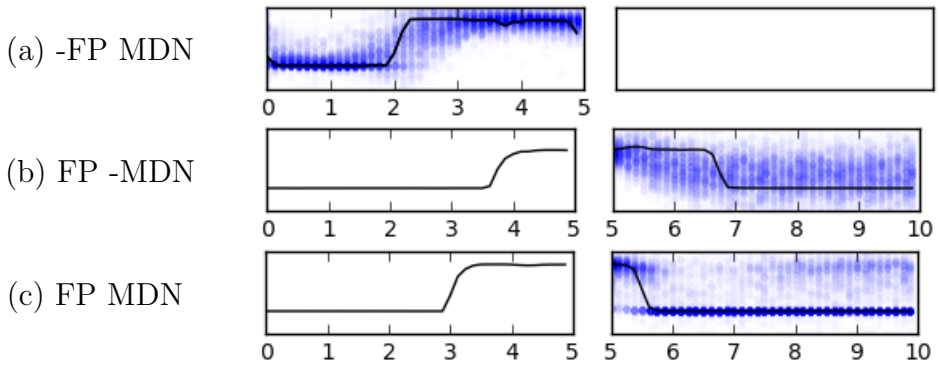


Figure 5.2: **Prediction visualization.** Inputs and ground truth (black) are shown along with predictions (blue). -FP MDN compresses and reconstructs the past; FP -MDN predicts one blurred future; and FP MDN predicts multiple possible futures.

5.4.2 Future Prediction

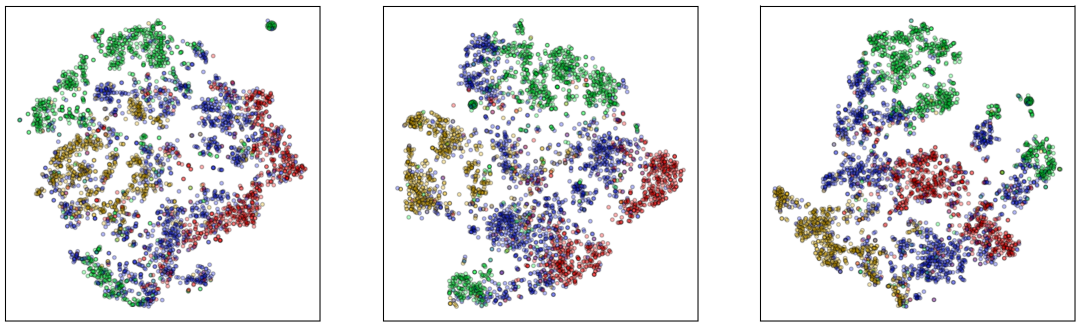
We train our model using 5 second windows of kinematics, extracted at random during training. Adam was used for optimization with a learning rate of 0.005, with other hyperparameters fixed to their defaults [84]. We trained for 5000 steps using a batch size of 50 (approximately 50 epochs). The hyperparameters tuned in our experiments were n_h, the number of hidden units for the encoder and decoder LSTMs, and n_c, the number of mixture components. For hyperparameter selection, 4 subjects were held out for validation. We began overly simple with $n_h = 16$ and $n_c = 1$, and proceeded to double n_h or n_c whenever doing so improved the held-out likelihood. This led to final values of $n_h = 64$ and $n_c = 16$.

Results for future prediction are shown in Figure 5.2. These predictions highlight the differences between baselines. On the top is an autoencoder-based approach, which simply reconstructs the past and does not predict the future. Next, we see future-prediction results for a model with only one mixture component, which cor-

responds to a unimodal Gaussian and therefore minimizing mean-squared-error loss. Here we see blurring among modes, with no ability to predict the future with any accuracy after only a few time steps. Finally, we see the main model of interest, which combines future prediction with mixture density networks to capture multimodal behavior. In this case, the model is confident that the signal will remain high (gripper open) and confident that it will then drop (gripper closed). After time passes, however, we see multiple modes arise, one corresponding to the gripper being closed and one corresponding to the gripper being open. Importantly, we see very little probability density in the middle state. This corresponds to the fact that little time is spent in this state, as it typically only occurs in between transitions.

5.4.3 Unsupervised Discovery of Activities

2-D visualizations of the learned representations, obtained with t-SNE [85], are shown in Figure 5.3. We again emphasize that the embeddings, and their t-SNE reduced representations, were formed without any knowledge of high-level activities. Upon coloring the points according to high-level activities, as in the figure, we can see that all three representation-learning methods lead to some *discovery* of the underlying activities. Notably, the autoencoder-based approach, -FP MDN, and the unimodal future-prediction approach, FP -MDN, both lead to less separation between activities. The full future-prediction model, FP MDN, leads to sharper separation. It is also interesting to note that *suture throw* (green), *knot tying* (orange), and *grasp*



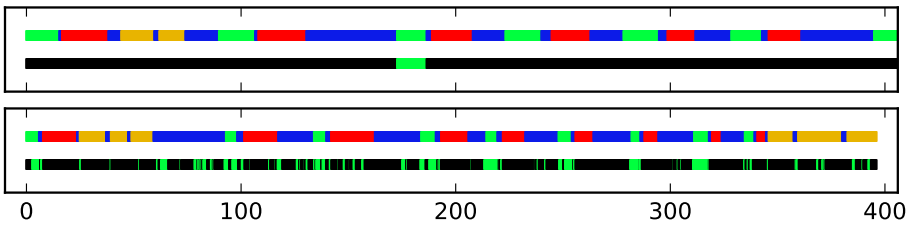
(a) -FP MDN (b) FP -MDN (c) FP MDN

Figure 5.3: **2-D dimensionality reductions, obtained using t-SNE, of the learned 64-D encodings, and colored according to activity.** The colors correspond to Suture Throw (green), Knot Tying (orange), Grasp Pull Run Suture (red), and Intermaneuver Segment (blue). We emphasize that the activity annotations were not used to obtain the encodings or their dimensionality-reduced versions; they are only used to color the resulting representations for visualization. Future prediction and MDNs both lead to more separation between high-level activities in the encoding space.

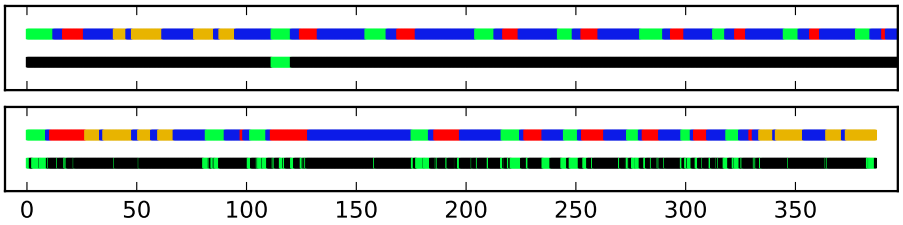
pull run suture (red) are all separated by *intermaneuver segment* (blue). This of course makes sense: *intermaneuver segment* was designated to encompass all activity between the other activities, which are those of primary interest.

5.4.4 Information Retrieval with Motion-Based Queries

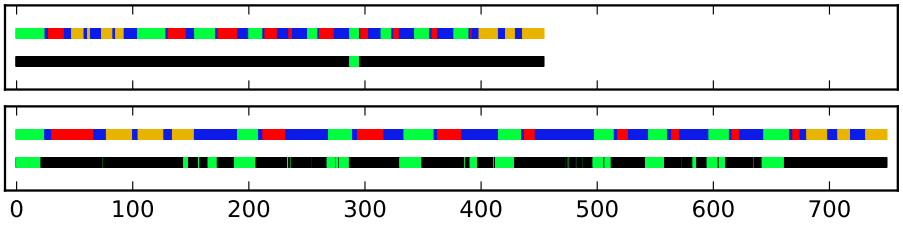
Next, we present results for retrieving kinematic frames based on a motion-based query, using the tasks of suturing and knot tying. We focus on the most difficult but most useful scenario: querying with a sequence from one subject i and retrieving frames from emphother subjects $j \neq i$. This setup resembles that of leave-one-user-out evaluation from Chapter 3, where we again focus on the scenario that is most



(a) Representative -FP MDN example. Precision: 0.47. Recall: 0.78. F1 Score: 0.59.



(b) Representative FP -MDN example. Precision: 0.54. Recall: 0.70. F1 Score: 0.61.



(c) Representative FP MDN example. Precision: 0.76. Recall: 0.78. F1 Score: 0.77.

Figure 5.4: **Qualitative results for kinematics-based suturing queries.** For each example, from top to bottom, we show 1) a full activity sequence from one subject; 2) the segment used as a query; 3) a full activity sequence from a *different* subject; and 4) the retrieved frames from our query. These examples were chosen because they exhibit precisions, recalls, and F1 scores that are close to the averages reported in Table 5.1.

relevant to practice: when the database of motion does not contain the subject that is performing the query.

In order to retrieve kinematic frames, we form encodings using *all windows* within one segment of an activity by subject i, compute the cosines between these encodings and all encodings for subject j, take the maximum (over windows) on a per-frame basis, and threshold. For evaluation, we follow [76], computing each metric (precision,

Table 5.1: **Quantitative results for motion-based queries from a database of surgical motion.** DAE + AS-DTW is the previous state-of-the-art approach; -FP MDN and FP -MDN are baselines without future prediction and mixture density networks, respectively; and FP MDN is the full model.

	Precision	Recall	F1 Score
		Suturing	
DAE + AS-DTW [76]	0.53 ± 0.15	0.75 ± 0.16	0.60 ± 0.14
-FP MDN	0.50 ± 0.08	0.75 ± 0.07	0.59 ± 0.07
FP -MDN	0.54 ± 0.07	0.76 ± 0.08	0.62 ± 0.06
FP MDN	0.81 ± 0.06	0.74 ± 0.10	$\mathbf{0.77 \pm 0.05}$
		Knot Tying	
DAE + AS-DTW [76]	—	—	—
-FP MDN	0.37 ± 0.05	0.73 ± 0.02	0.49 ± 0.05
FP -MDN	0.34 ± 0.05	0.74 ± 0.02	0.46 ± 0.05
FP MDN	0.62 ± 0.08	0.74 ± 0.04	$\mathbf{0.67 \pm 0.05}$

recall, and F1 score) from each source subject i to each target subject $j \neq i$, and finally averaging over all target subjects.

Quantitative results are shown in Table 5.1, comparing the FP MDN to its baselines and the state-of-the-art approach [76], and qualitative results are shown in Figure 5.4. We can see that the FP MDN significantly outperforms the two simpler baselines, as well as the state-of-the-art approach in the case of suturing, improving from an F1 score of 0.60 ± 0.14 to 0.77 ± 0.05.

5.5 A Generative Future-Prediction Model

Here we forego the distinction between previous and future windows of kinematics. Instead, let $\mathbf{X} \equiv \{\mathbf{x}_t\}_1^T$ denote a full sequence of kinematics, with each $\mathbf{x}_t \in \mathbb{R}^{n_x}$ (containing, e.g., joint velocities). We model the full joint distribution over $\mathbf{X}$ by making use of the chain rule,

$$p(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T) = p(\mathbf{x}_1)p(\mathbf{x}_2 \mid \mathbf{x}_1)p(\mathbf{x}_3 \mid \mathbf{x}_1, \mathbf{x}_2) \cdots p(\mathbf{x}_T \mid \mathbf{x}_1, \ldots, \mathbf{x}_{T-1}) \tag{5.7}$$

Here, we will model each factor in the above product sequentially: At time t, we use long short-term memory (LSTM) [32,45] to map from the previous kinematics vector $\mathbf{x}_{t-1}$ and previous hidden state $\mathbf{h}_{t-1}$ to a new hidden state via Equation 5.1. Next, we map from $\mathbf{h}_t$ to the parameters that govern the distribution $p(\mathbf{x}_t \mid \mathbf{x}_1, \ldots, \mathbf{x}_{t-1})$. Thus, for each time step, we have

$$\mathbf{h}_t = \text{LSTM}(\mathbf{x}_{t-1}, \mathbf{h}_{t-1}) \tag{5.8}$$

$$\boldsymbol{\pi}_t = \text{softmax}(\mathbf{W}_\pi \mathbf{h}_t + \mathbf{b}_\pi) \tag{5.9}$$

$$\boldsymbol{\mu}_t^{(c)} = \mathbf{W}_\mu^{(c)} \mathbf{h}_t + \mathbf{b}_\mu^{(c)} \tag{5.10}$$

$$\mathbf{v}_t^{(c)} = \text{softplus}(\mathbf{W}_v^{(c)} \mathbf{h}_t + \mathbf{b}_v^{(c)}) \tag{5.11}$$

with the conditional distribution over $\mathbf{x}_t$ then specified as

$$p(\mathbf{x}_t \mid \mathbf{x}_1, \ldots, \mathbf{x}_{t-1}) = \sum_c \pi_t^{(c)} \mathcal{N}\Big(\mathbf{x}_t\,;\ \boldsymbol{\mu}_t^{(c)}, \mathbf{v}_t^{(c)}\Big) \tag{5.12}$$

Note that, unlike the model in Section 5.3, independence is *not* assumed over time, and only one LSTM network is used (rather than one encoder LSTM and one decoder LSTM). All weight matrices $\mathbf{W}$ and all biases $\mathbf{b}$, in both the LSTM and in the mapping from hidden states to distribution parameters, are learned by maximizing (the logarithm of) Eq. 5.7. We remark that some care must be taken in order to maintain numerical stability to optimize this objective; please see the Appendix for details.

5.6 Experiments: Data-Efficient Activity Recognition

In this section our goal is data-efficient activity recognition. This will be carried out for varying amounts of available training data, and will consist of two phases: 1. unsupervised representation learning and 2. leveraging the learned representations during the recognition phase.

During the unsupervised representation phase, we aim to transform any given sequence $\mathbf{X}$ to a new sequence $\tilde{\mathbf{X}}$. Four representations are considered. The first

are the raw inputs themselves, such that $\tilde{\mathbf{X}} = \mathbf{X}$. This corresponds to the same recognition approach that was considered in Chapter 3, though here the availability of less annotated data will be considered. The second, third, and fourth approaches were described previously in this chapter: an autoencoder-based approach, which learns to reconstruct a window of motion through a bottleneck; a window-based future-prediction approach, which learns to predict an incoherent window of future motion from a past window; and the generative approach, which does not assume fixed window sizes or independence over time.

For recognition, we proceed in a manner similar to that of Chapter 3, modeling the conditional distribution over the label sequence $\mathbf{Y} \equiv y_1, y_2, \ldots, y_T$ that we desire. The key difference is that we model $p(\mathbf{Y} \mid \tilde{\mathbf{X}})$, where the learned representations $\tilde{\mathbf{X}}$ are presumably more amenable to annotation-scarce activity recognition than the raw inputs themselves. In three respresentation-learning scenarios, $\tilde{\mathbf{X}}$ is taken to be the learned internal hidden states of the LSTM: $\tilde{\mathbf{X}} = \mathbf{H} \equiv \mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_T$. Importantly, for recognition, we use the state-of-the-art approach from Chapter 3, which consists of a multilayered, bidirectional LSTM architecture. Training is carried out by maximizing conditional likelihood under this model, or equivalently by minimizing cross entropy. Hyperparameters are carried over directly, and so any gains in performance from using the learned representations are not the result of hyperparameter tuning.

5.6.1 Datasets

We consider data-scarce recognition both in the context of maneuver recognition and gesture recognition. For maneuver recognition, we use the Minimally Invasive Surgical Training and Innovation Center – Science of Learning dataset (MISTIC-SL) [42,76], and for gesture recognition, we use the JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS) [19,41]. In the case of maneuver recognition, each time step is labeled as 1 of 4 different manuevers: *suture throw* (ST), *knot tying* (KT), *grasp pull run suture* (GPRS), or *intermaneuver segment* (IMS). In the case of gesture recognition, each time step is labeled as 1 of 10 different low-level gestures. In both cases, the raw inputs consist of the same 14 kinematic signals as described in Chapter 3; and again all signals are downsampled by a factor of 6, as in previous work.

5.6.2 Experimental Design

The recognition model from Chapter 3 consists of bidirectional LSTM with 3 layers, each with 64 hidden units, and is optimized using Adam and a learning rate of $10^{-2.5}$. This state-of-the-art model, along with all hyperparameters except batch size, are carried over unchanged. We use a batch size of 1 because it is the only possible option in many of the experiments. Training is carried out for 100 epochs. The metrics considered are frame-wise error rate and segment-wise edit distance (Levenshtein distance), normalized by the maximum number of segments in any one trial to aid

interpretability, following prior work. In focusing on generalization across users, *any particular training set consists of exactly one labeled trial per user*, with between 1 and $u-1$ labeled trials, where u is the number of users in the dataset. In all cases, results are averaged over splits, exhaustively for 1 trial and $u-1$ trials and randomly otherwise (in this case 10 splits are randomly sampled).

During the representation-learning phase, the most important hyperparameters are the number of LSTM hidden units (n_h) and the number of components in the Gaussian mixture model (n_c). The autoencoder and future-prediction models were tuned above to maximize performance on a held-out set of 4 MISTIC-SL users, and here we use the same values ($n_h = 64, n_c = 16$). We followed the same process for the RNN-Based Generative Model. This led to values of $n_h = 128, n_c = 8$. In all cases, training was carried out for 100 epochs using Adam, with a learning rate of 0.005. We emphasize that in all cases, tuning was carried out using the unsupervised-learning objective, not the recognition objective.

5.6.3 Future Prediction

Figure 5.5 shows examples predictions using the gripper-angle signal from MISTIC-SL. This illustrates the major differences between the three methods considered for unsupervised representation learning. The first two were considered in Section 5.4; on top, we see an autoencoder-based approach, and in the middle, we see blurry, window-based future predictions. On the bottom, we see results from the generative

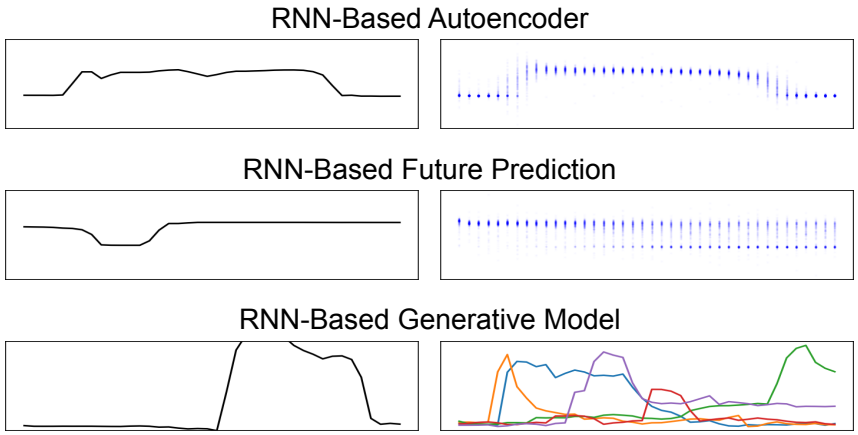


Figure 5.5: **Example predictions for the three tasks considered for unsupervised representation learning.** On the left, we see the input to each model; and on the right, we see sampled predictions. The autoencoder reconstructs the input window; the future-prediction model predicts a window assuming independence over time steps, conditioned on a previous window; and the generative model predicts coherent futures of any length, conditioned on the entire past (5 sampled trajectories are shown).

model introduced in Section 5.5. In particular, this model is capable of predicting coherent trajectories. This suggests that the generative model's representations may be better suited for fine-grained activities; and we will see that this is confirmed below in the case of gesture recognition using the JIGSAWS dataset.

5.6.4 Maneuver Recognition with Scarce Annotations

Figure 5.6 shows error rate vs. the number of labeled trials for maneuver recognition (MISTIC-SL). Using only one labeled trial, raw inputs for recognition lead to an error rate of 27.3%; the autoencoder representations lead to an improvement,

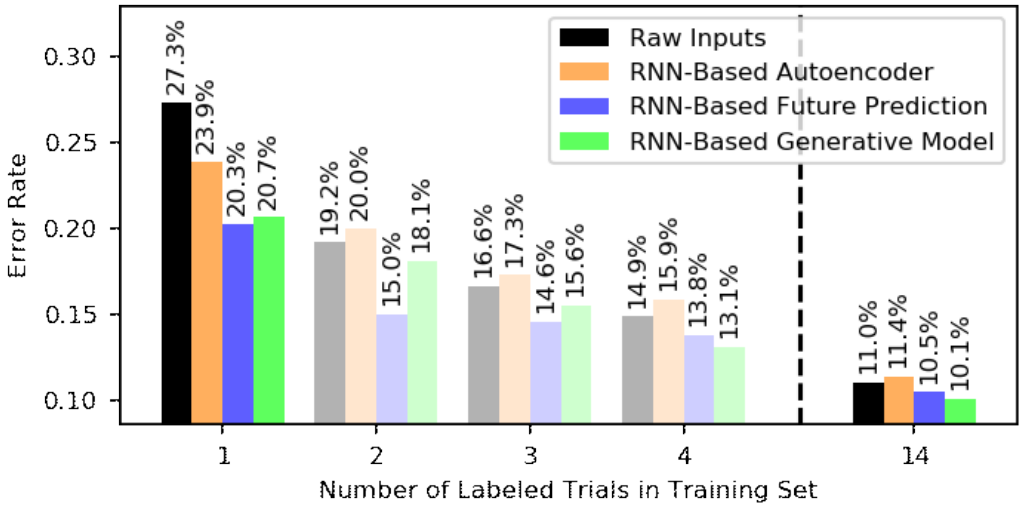


Figure 5.6: **MISTIC-SL Maneuver Recognition: Error rate vs. number of labeled trials.** The bottom of the y axis is set to 8.7%, the best published result using LSTM (∼36 labeled trials). The non-transparent results are reported over exhaustive, deterministic splits (see Section 3.5) which can be compared to in future work.

obtaining 23.9%; and the future-prediction and generative models lead to further improvements, obtaining 20.3% and 20.7% respectively. Figure 5.8 shows qualitative results using only 1 labeled trial with future prediction (results are similar for the generative model). When 14 labeled trials are used, the generative model yields the lowest error rate (10.1%). For reference, the state-of-the-art LSTM result using 36 labeled trials is 8.7% [86]. Also included is an example qualitative result, in Figure 5.8. We remark that the predictions using both future-prediction approaches look similar. In addition, as in Chapter 3, we include edit distances for each approach, again as a function of the amount of available training data. These results are shown in Figure 5.7. In general, the same trends that we have seen for error rates carry over

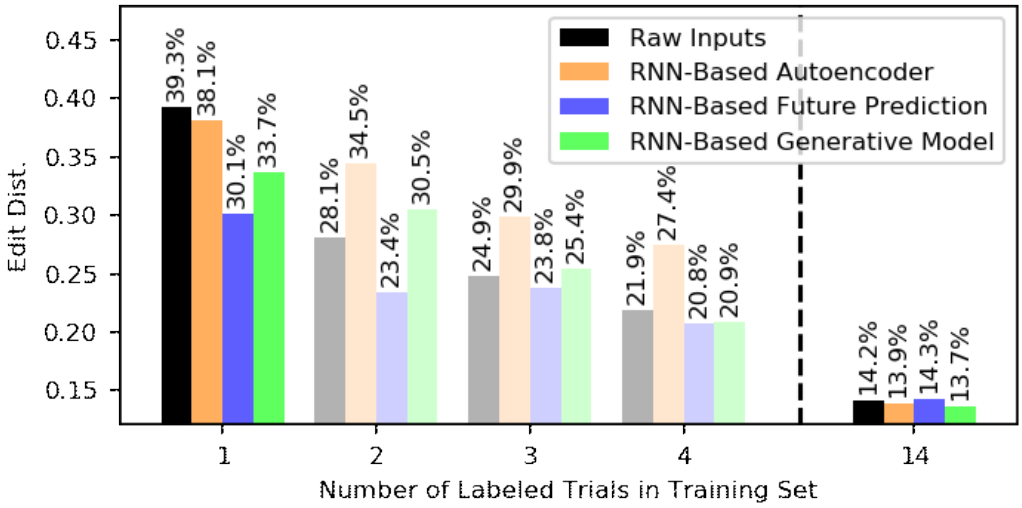


Figure 5.7: **MISTIC-SL Maneuver Recognition: Edit distance vs. number of labeled trials.** The bottom of the y axis is set to 12.1%, the best published result using LSTM (∼36 labeled trials). The non-transparent results are reported over exhaustive, deterministic splits (see Section 3.5) which can be compared to in future work.

to edit distance as well.

5.6.5 Gesture Recognition with Scarce Annotations

Figure 5.9 shows error rate vs. the number of labeled trials for gesture recognition (JIGSAWS). Using only one labeled trial, raw inputs lead to an error rate of 33.6%. Representations from the full generative model reduce the error rate to 29.6%, while the autoencoder and future-prediction based representations both degrade performance. This is not surprising: we have no reason to believe that the autoencoder's task of signal reconstruction is well aligned with the task of activity recognition; and for future prediction, we have seen above that blurry, uncoherent futures are obtained,

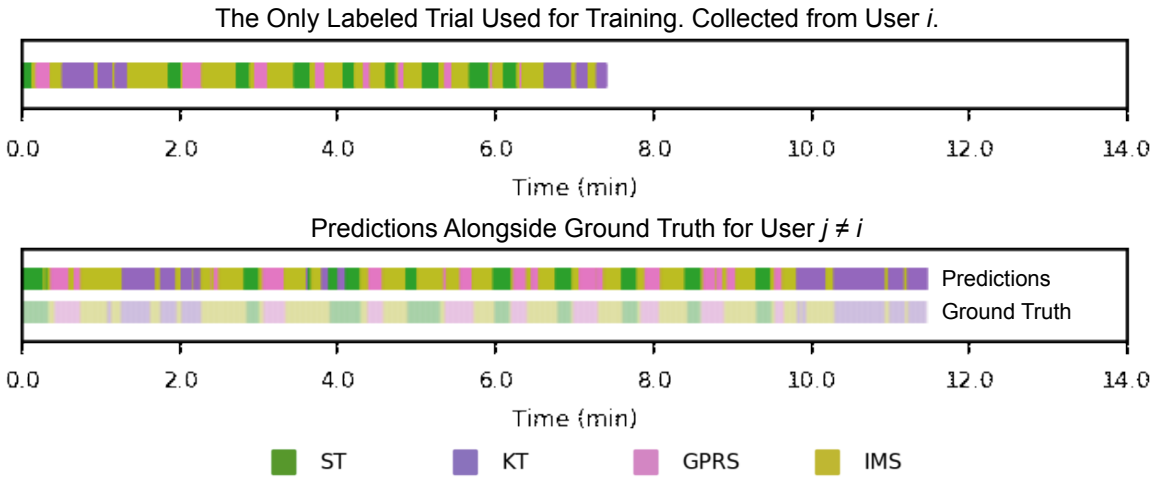


Figure 5.8: **Example predictions for maneuver recognition, using only a single labeled sequence for training.** Here representations were learned with the RNN-Based Future Prediction model prior to recognition. It exhibits a representative error rate (19.4%) and an edit distance that is worse than average (40.7%). Results are similar for the RNN-Based Generative Model. The activities are *suture throw* (ST), *knot tying* (KT), *grasp pull run suture* (GPRS), and *intermaneuver segment* (IMS).

and this is likely detrimental to recognizing fine-grained activities such as gestures. When 7 labeled trials are used for training, the generative model again yields the lowest error rate (17.6%). For reference, the state-of-the-art LSTM result using 35 labeled trials is 15.3% [86]. Figure 5.10 shows results for edit distance; the same general trends hold.

5.7 Additional Experiments

This chapter is primarily concerned with unsupervised representation learning and data-efficient recognition in the context of surgical activity recognition. However, an

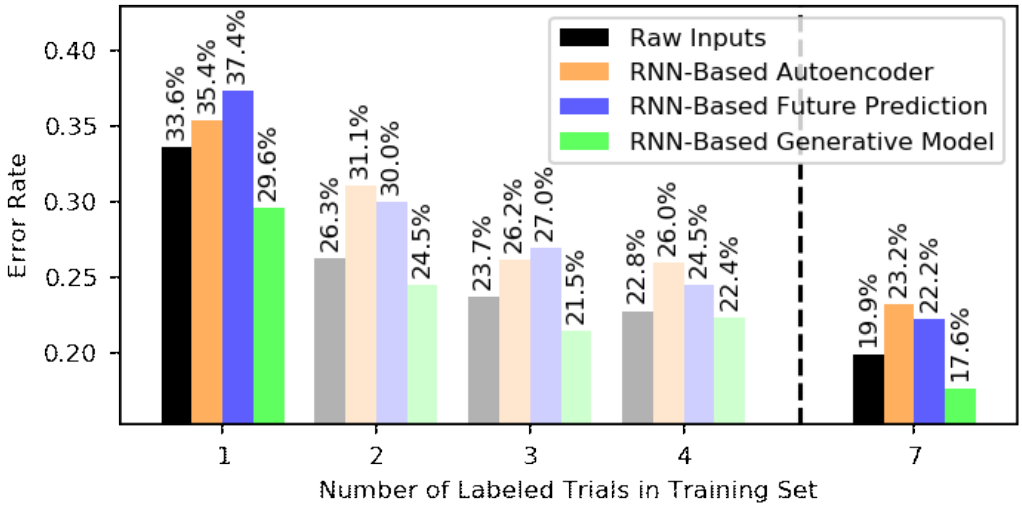


Figure 5.9: **JIGSAWS Gesture Recognition: Error rate vs. number of labeled trials.** The bottom of the y axis is set to 15.3%, the best published result using LSTM (∼35 labeled trials). The non-transparent results are reported over exhaustive, deterministic splits (see Section 3.5) which can be compared to in future work.

interesting question is whether the trends that we've seen carry over to other tasks. Above, we found that future prediction can lead to large performance gains when annotated data is limited, especially using the RNN-based generative model; and that autoencoder-based representations are less amenable to data-efficient learning. In this section, we consider three additional experiments to investigate these findings further. The first two experiments involve isolated simulations that are free from the many complications that arise in real-world data; and in the third experiment, we consider phoneme recognition from speech data.

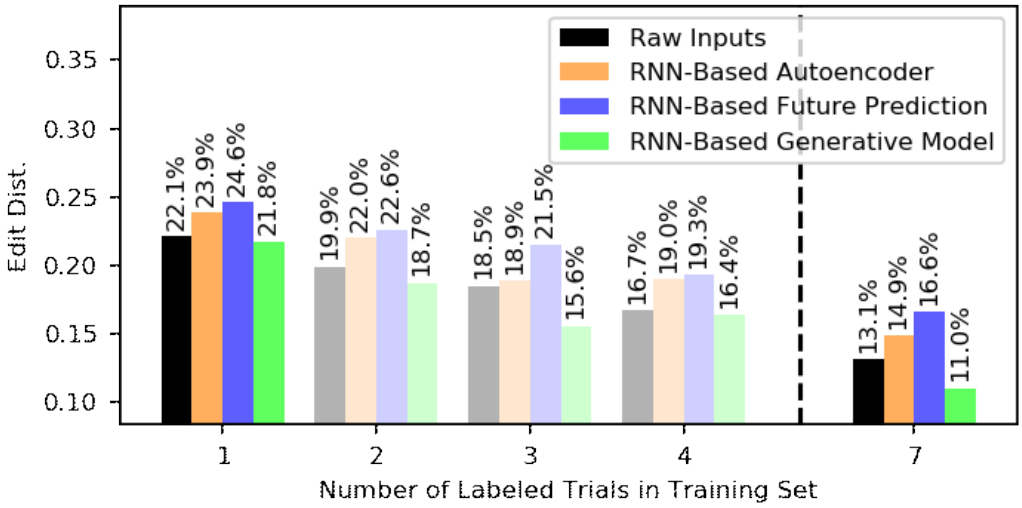


Figure 5.10: **JIGSAWS Gesture Recognition: Edit distance vs. number of labeled trials.** The bottom of the y axis is set to 8.4%, the best published result using LSTM (∼35 labeled trials). The non-transparent results are reported over exhaustive, deterministic splits (see Section 3.5) which can be compared to in future work.

5.7.1 Discovering the Modes of a Constrained Pendulum

We first consider a pendulum constrained by two pins (Figures 5.11 and 5.12), a system which has three modes: contact with no pin, contact with the left pin, or contact with the right pin. Here, the goal is to *discover* these modes in an unsupervised fashion from state observations alone.

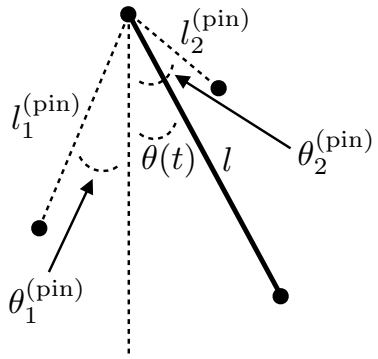


Figure 5.11: **A pendulum constrained by two pins.** The pendulum exhibits three modes: contact with no pin (as in this figure), contact with the left pin (as in Figure 5.12), and contact with the right pin.

When the pendulum is not in contact with any pin, it is governed by the system

$$\dot{\theta} = \frac{v}{l} \tag{5.13}$$

$$\dot{v} = -g \sin\theta \tag{5.14}$$

where θ is the angle from the pendulum's rest position; v is the linear velocity at the end of the pendulum; g is gravity; and $\dot{\theta}$ and $\dot{v}$ are the first order derivatives of θ and v with respect to time. When the pendulum contacts a pin, it remains governed by the same system, but with l replaced by l', the distance from the contacted pin to the end of the pendulum (see Figure 5.12). Here the state of the system is taken to be $[\theta, v]$ because it avoids discontinuities in state at the moments of pin contact; for more detail, see [87].

Of course, when the positions of the pins ($l_1^{(\text{pin})}$, $\theta_1^{(\text{pin})}$, $l_2^{(\text{pin})}$, and $\theta_2^{(\text{pin})}$ in Figure 5.11) are fixed, then the modes are perfectly separable by simply examining θ, and

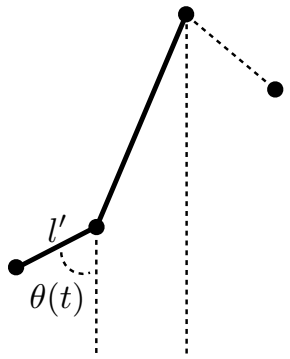
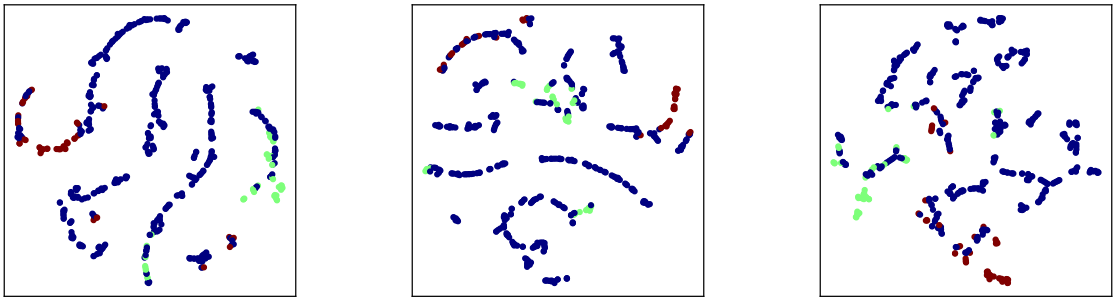


Figure 5.12: **A pendulum constrained by two pins, as in Figure 5.11, but here operating in a constrained mode (making contact with the pin on the left).** When the pendulum contacts the pin, it remains governed by the same differential equations, but with the length of the pendulum, l, replaced by its distance from the pin, $l' = l - l_1^{(\text{pin})}$.

the task of mode discovery is trivial. Instead, we consider a scenario in which pins are placed randomly during each observation. In our experiments, the pins are placed symmetrically about the pendulum's rest position, and the magnitude of the angles is sampled uniformly at random from an interval spanning $\pi/4$ to $\pi/2$ radians. All LSTM hidden states and encodings consist of a small number of $n_h = 8$ hidden units, and where applicable, $n_c = 2$ components are used in mixture models. We optimize using the Adam optimizer [84] with a learning rate of 0.005 and a batch size of 128 for 10,000 steps, which allows for convergence in all cases.

2-D visualizations obtained with t-SNE [85] are shown in Figure 5.13, using observations sequences that were never seen during training. Each point corresponds to a single time step, and is colored according to its mode at that time step: in contact with no pin (blue), in contact with the pin in the left (green), or in contact



(a) Autoencoder

(b) Window-Based Future Prediction

(c) Generative Future Prediction

Figure 5.13: **Dimensionality reduced visualizations (via t-SNE) of the representations obtained using motion data from a constrained pendulum obstructed by two pins, with three underlying modes (Figures 5.11 and 5.12).** The points are colored according to its mode at each time step.

with the pin on the right (red). Qualitatively, Figure 5.13 suggests that the three representation-learning methods perform similarly.

One can gain only so much insight from 2-D visualizations of the 8-D embedding space. In addition, we examined the problem of mode classification using linear decision boundaries, as a proxy for how well we were able to discover the modes. In this case, a random-chance classifier would yield an error rate of 66.6%. Results under various amounts of training data are shown in Table 5.2. Overall, when little annotated data is available, we see that all representations yield similar performance. As the amount of labeled data becomes more abundant, the future-prediction-based methods lead to better mode discovery.

As mentioned above, the pins are placed randomly for each observation sequence, as otherwise simply thresholding θ would specify the modes deterministically. We

Table 5.2: **Error rates for classifying the modes of a constrained pendulum, under various amounts of available labeled training data.** Original representations (pendulum state) are considered alongside three learned representations. Each entry is averaged over 10 runs, where each run corresponds to a different random subset of 16,384 sequences.

# Labeled Points	Orig. Representations	Autoencoder	Window-Based FP	Generative FP
100	20.0%	21.2%	**19.5**%	19.8%
200	**18.9**%	20.0%	19.6%	19.7%
400	19.0%	19.9%	**18.1**%	19.5%
800	17.8%	18.8%	**17.4**%	17.7%
5000	16.5%	16.9%	13.8%	**13.3**%

attribute nearly equal performance at low-annotation counts to a similar issue: although pins are placed randomly, the instantaneous pendulum state alone – without any dynamics – leads to a high degree of mode separation. In our next experiment, we design an experiment to specifically alleviate this issue.

5.7.2 Discovering the Modes of a Driven Pendulum

For the constrained pendulums of the previous section, the state vector is highly correlated with the resulting modes. We suspect that this is why all three representations learning methods – autoencoders, window-based future prediction, and the full generative model – all yield similar accuracies for mode discovery. Intuitively, any projection of θ and $\dot{v}$ into higher dimensional space will likely leave these modes

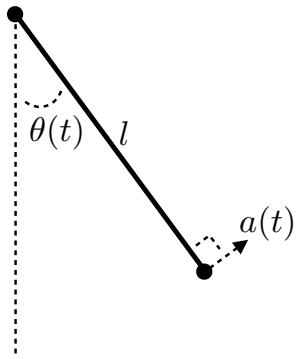
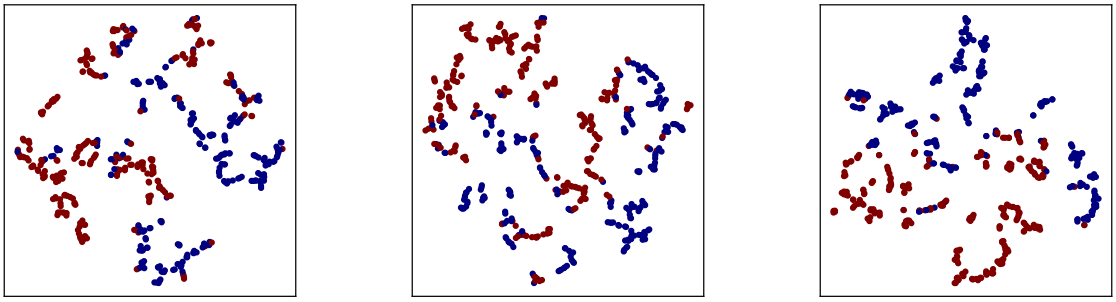


Figure 5.14: **An unconstrained pendulum with acceleration applied along its axis of travel (perpendicular to the direction defined by the string).** The magnitude of $a(t)$ is constant, but its sign changes after random intervals of time. Thus the system exhibits two modes. Unlike the constrained-pendulum experiments, these modes are uncorrelated with the pendulum's state vector, which here is $[\theta(t), \dot{\theta}(t)]$.

separable.

In this section, we design an experiment in which the state vectors themselves are uncorrelated with the underlying modes. We consider an unconstrained pendulum with an acceleration unit attached to the end of the pendulum. Acceleration is always applied along the axis of travel, or in other words perpendicular to the direction of the pendulum's string (see Figure 5.14). In addition, the magnitude of the applied acceleration is fixed, but its sign flips randomly, which serves as the underlying system mode that we wish to discover. In our experiments, each observed sequence is two seconds long, with fixed sign intervals chosen uniformly at random, between 0.25 seconds and 0.5 seconds. Once again, all LSTM hidden states and encodings consist of a small number of $n_h = 8$ hidden units, and where applicable, $n_c = 2$ components are used in mixture models. We optimize using the Adam optimizer [84] with a



(a) Autoencoder

(b) Window-Based Future Prediction

(c) Generative Future Prediction

Figure 5.15: **2-D t-SNE visualizations of the representations obtained using motion data from a pendulum with time-varying force applied along the axis of travel (see Figure 5.14).** The points are colored according to its discrete mode at each time step, which determines the sign of the applied force. Qualitatively, we can see that generative future prediction yields representations that discover these modes with higher fidelity. The same also holds quantitatively: a single separating hyperplane fit in the original (non-reduced) embedding spaces yields accuracies of 75.0% when using an autoencoder, 92.2% when using window-based future prediction; and 95.9% when using generative future prediction.

learning rate of 0.005 and a batch size of 128 for 10,000 steps, which allows for convergence in all cases.

As in the previous section, 2-D visualizations obtained with t-SNE [85] are shown in Figure 5.15. Here, all respresentation-learning methods yield visualizations with reasonable separation, though the full generative model for future prediction does seem to better separate the modes.

Once again, we consider performance using linear decision boundaries as a proxy for successful mode separation. Results for varying amounts of training data, using logistic regression, are shown in Table 5.3. Here, as expected, using the instantaneous pendulum states leads to nearly no ability to distinguish between modes:

Table 5.3: **Error rates for classifying the modes of a pendulum with time-varying forced applied along the axis of travel, under various amounts of available labeled training data.** Original representations (pendulum state) are considered alongside three learned representations. Each entry is averaged over 10 runs, where each run corresponds to a different random subset of 16,384 sequences.

# Labeled Points	Orig. Representations	Autoencoder	Window-Based FP	Generative FP
100	47.1%	29.6%	19.8%	**4.2**%
200	50.1%	28.0%	15.2%	**3.6**%
400	48.6%	27.0%	11.8%	**3.3**%
800	49.9%	26.0%	10.1%	**3.0**%
5000	46.0%	24.2%	7.4 %	**2.5**%

even with 5,000 labeled points, we achieve 46.0% error, whereas a random-chance classifier would achieve 50.0% error. All representation-learning techniques lead to representations that better distinguish between modes. However, here we see stark contrast among representation-learning methods. The autoencoder performs considerably better than random chance, but achieves approximately 24.2% error even with 5,000 labeled points. Meanwhile, window-based future prediction yields an error rate of 19.8% with between 2 and 3 orders of magnitude less annotated data (100 labeled points); and generative future prediction goes even further, achieving 4.2% error with only 100 labeled points.

In the section that follows, we consider phoneme recognition from speech data. We remark that, like the simulated experiment in this section, phoneme recognition requires dynamics in order to distinguish between the underyling system modes (phonemes). In other words, the observation vector at any moment is essentially

Table 5.4: **Frame-wise error rates for recognition of phonemes (TIMIT) from speech, under various amounts of available labeled training data.** Original (MFCC) representations are considered alongside three learned representations. Each entry is averaged over three runs, where each run corresponds to a different random subset of the original full training set of 3696 sequences. In all cases, the standard deviation over these 3 runs is less than 0.6%.

# Labeled Seqs.	Orig. Representations	Autoencoder	Window-Based FP	Generative FP
100	50.1%	52.1%	45.6%	**41.4**%
250	45.7%	47.7%	42.5%	**38.2**%
500	42.4%	44.1%	40.2%	**36.3**%
1000	39.0%	40.9%	37.6%	**34.7**%
3696	32.6%	35.1%	33.8%	**31.7**%

useless in isolation.

5.7.3 Data-Efficient Phoneme Recognition

Here we combine future prediction and LSTM-based recognition for online, frame-wise phoneme recognition, using the TIMIT corpus [67]. The standard splits are based on [69], which includes 3696 sequences for training, 400 for validation, and 192 for testing. Each frame of every sequence is associated with 1 of 61 phonemes. However, following common practice, as in [68], we collapse this set into 39 phonemes with less overlap, and include glottal stops, for a total of 40 classes in total. We follow [34] and extract 12 mel frequency cepstral coefficients plus energy using 25 ms Hamming windows and a pre-emphasis coefficient of 0.97. This results in 13 inputs per frame, and each input is normalized on a per-sequence basis to have a mean of 0 and a vari-

ance of 1. After representation learning, recognition is carried out using a 1 LSTM layer with 128 hidden units, optimized with Adam using a learning rate of 0.001 for 15 epochs.

Table 5.4 shows frame-wise error rates obtained when varying amounts of training data are available, and under varying representations of the input sequences. Each error rate is averaged over three runs, where the runs correspond to different subsets of labeled sequences (sampled at random from the original set of 3696 sequences). An immediate observation is that, regardless of representation, error rates continue to decrease with more labeled data, even at the full training set of 3696 sequences. This suggests that collecting even more densely-annotated sequences would continue to improve performance. With regard to significance, we note that the standard deviation corresponding to any individual error rate is at most 0.6%.

When few training sequences are available for training, learned representations from future prediction lead to significant gains in performance. For example, when 100 labeled sequences are available, the original MFCC representations lead to an error rate of 50.1%; autoencoder-based representations lead to 52.1% error; and the window-based and generative future-prediction representations lead to 45.6% error and 41.4% error, respectively. For comparison, the original MFCC representations yields 45.7% error when 250 labeled sequences are available, and 42.4% when 500 labeled sequences are available. In other words, future-prediction-based representation learning lets us achieve 42% error with 5x fewer annotations.

In the case that all 3696 training sequences are available for training, using the original MFCC representations leads to an error rate of 32.6%; autoencoder-based representations lead to 35.1% error; window-based future-prediction based representations lead to 33.8% error; and representations based on generative future prediction lead to 31.7% error. Relative performance is somewhat similar to when fewer annotated sequences are available, but with narrower gaps, and with window-based future prediction leading to representations that are now outperformed by the original MFCC representations. This is not a surprising result: unlike generative future prediction, window-based future prediction leads to blurry, incoherent futures, and we must expect that, given enough annotated data, representations can be learned without such signal loss. We observed a similar phenomenon in the case of gesture recognition, in which fine details (over short time periods) are important (see Section 3.7).

Finally, recall that all results were computed using the same hyperparameters: 1 LSTM layer with 128 hidden units, optimized with Adam using a learning rate of 0.001 for 15 epochs. Is it possible that different hyperparameters may allow the original MFCC representations to narrow the large performance gap, especially in the scarce-annotation regime? To answer this question we focus on the scenario in which only 100 labeled sequences are available for training, since this allows for quick experimentation. We evaluated performance using raw representations over a full grid search, with 1 or 2 LSTM layers; 64, 128, or 256 hidden units per layer; and a

learning rate of 0.01, 0.001, or 0.0001. In addition, we trained for 25 epochs (instead of 15) and we allowed for an extreme version of early stopping, where the *best* test performance at any point in training is taken to be the final performance. Even in this overly-optimistic setting, original MFCC representations do not surpass an error rate of 47.0%, which is still well behind our original future-prediction based results (45.6% for window-based prediction, and 41.4% using the full generative model).

5.8 Conclusions

In practice, obtaining manual annotations is difficult, expensive, and error-prone – especially when carried out at scale. However, essentially all prior work toward automated surgical activity recognition assumed an abundance of densely-annotated sequences for training. In this chapter, we introduced future-prediction based models for unsupervised representation learning; showed that these methods are capable of *discovering* high-level surgical activities; and demonstrated that these methods can be used in downstream tasks such as surgical activity recognition for improved performance in the scarce-annotation regime. The RNN-based generative model introduced here is particularly strong for this purpose, especially when recognizing activities that occur over short time scales, as in gesture recognition. That said, a significant gap still exists in performances when fewer annotated sequences are available for training. We hope that the community will join us in seeing how much we can improve

performance in this important annotation-limited regime.

5.9 Appendix

We remarked that in practice we actually maximize the logarithm of the likelihood, rather than the likelihood itself, for numerical stability. In order to obtain reasonable results, it is necessary to proceed with care, and to use a common trick for computing the logarithm of a sum of exponentials in a numerically-stable fashion. Here we describe this trick for completeness.

Computing

$$f(\mathbf{x}) = \log\Big(\sum_i \exp(x_i)\Big)$$

will fail numerically if the x_i are very small, because all of the terms in the sum will be very close to 0 (or equal to 0). We can fix this by shifting all of the x_i by some constant s:

$$\begin{aligned} f(\mathbf{x}) &= \log\Big(\sum_i \exp(x_i + s - s)\Big) \\ &= \log\Big(\exp(s)\sum_i \exp(x_i - s)\Big) \\ &= s + \log\Big(\sum_i \exp(x_i - s)\Big) \end{aligned}$$

A common choice is to let $s = \max_i x_i$; then the largest x_i maps to 0 in log space, or 1 after we exponentiate. Some small x_i may still disappear, but this is okay because

their contributions to the sum are negligible compared to the largest x_i's contribution.

For example, consider an array of length 1, $x_1 = -1000$. If we exponentiate this directly using 64-bit precision, we get 0, and therefore $\log \exp x_1 = -\infty$. Instead, if we use the above result with $s = -1000$, we obtain $\log \exp x_1 = -1000 + \log \exp 0 = -1000$.

Chapter 6

Conclusions

This thesis focused on automated surgical activity recognition, a foundation for many research opportunities that strive for more objective, more localized, and more abundant performance assessment and feedback during surgical training.

In Chapter 3, we introduced recurrent neural networks for the task of surgical activity recognition, and we demonstrated for the first time that it is possible to recognize activities at the granularity of *maneuvers*. Unlike the activities considered in prior work, maneuvers are already present in surgical training curricula and therefore already familiar to surgeons. In addition, we departed from the common path of focusing on probabilistic structure over activites, and instead focused on learning powerful unary terms, which are computed from observations themselves rather than from interdependencies among activities. Without any probabilistic structure over activities, this led to state-of-the-art performance for both gesture recognition and

maneuver recognition, in the latter case halving the error of the next-best method.

In Chapter 4, we investigated the necessity of capturing long-term dependencies for surgical activity recognition. Here we analyzed RNNs, and in particular NARX RNNs, in detail, and we developed a new recurrent-neural-network architecture, MIST RNNs, which 1. from a theoretical perspective can learn much longer dependencies that common architectures such as long short-term memory (LSTM) and 2. from an empirical perspective was capable of outperforming the most widely-used RNN architecture, LSTM, on such tasks. In the context of surgical activity recognitions, we found that MIST RNNs were able to rival LSTM from the perspective of error rate, but that MIST RNNs did not outperform LSTM, and so we conclude that short- to medium-length dynamics are most crucial for the tasks of gesture and maneuver recognition.

In Chapter 5, we asked whether it is possible to learn meaningful and useful representations of surgical motion *without* activity annotations, and whether it is possible to leverage these representations to improve performance in the downstream task of activity recognition. First, we showed that future prediction can be used to learn meaningful representations that lead to the *discovery* high-level activities (maneuvers). Next, we showed that these representations can be used to achieve state-of-the-art performance for querying a database of surgical motion with motion-based queries. And finally, we found that future-prediction based representation learning, prior to the recognition phase, yields significant performance gains for activity recog-

nition when few annotated sequence are available.

Bibliography

[1] J. D. Birkmeyer, J. F. Finks, A. O'reilly, M. Oerline, A. M. Carlin, A. R. Nunn, J. Dimick, M. Banerjee, and N. J. Birkmeyer, "Surgical skill and complication rates after bariatric surgery," *New England Journal of Medicine*, vol. 369, no. 15, pp. 1434–1442, 2013.

[2] E. Wenghofer, D. Klass, M. Abrahamowicz, D. Dauphinee, A. Jacques, S. Smee, D. Blackmore, N. Winslade, K. Reidel, I. Bartman, and R. Tamblyn, "Doctor scores on national qualifying examinations predict quality of care in future practice," *Medical education*, vol. 43, no. 12, pp. 1166–1173, 2009.

[3] K. A. Ericsson, "Deliberate practice and the acquisition and maintenance of expert performance in medicine and related domains," *Academic medicine*, vol. 79, no. 10, pp. S70–S81, 2004.

[4] S. S. Vedula, M. Ishii, and G. D. Hager, "Objective assessment of surgical technical skill and competency in the operating room," *Annual review of biomedical engineering*, vol. 19, pp. 301–325, 2017.

BIBLIOGRAPHY

[5] R. H. Bell, "Why johnny cannot operate," *Surgery*, vol. 146, no. 4, pp. 533–542, 2009.

[6] S. L. Gearhart, M.-H. Wang, M. M. Gilson, B. Chen, and D. E. Kern, "Teaching and assessing technical proficiency in surgical subspecialty fellowships," *Journal of surgical education*, vol. 69, no. 4, pp. 521–528, 2012.

[7] D. M. Jacobs and D. Poenaru, Eds., *Surgical Educators' Handbook*. Association for Surgical Education, 2001.

[8] D. J. Scott, J. C. Cendan, C. M. Pugh, R. M. Minter, G. L. Dunnington, and R. A. Kozar, "The changing face of surgical education: simulation as the new paradigm," *Journal of Surgical Research*, vol. 147, no. 2, pp. 189–193, 2008.

[9] M. Curry, A. Malpani, R. Li, T. Tantillo, A. Jog, R. Blanco, P. K. Ha, J. Califano, R. Kumar, and J. Richmon, "Objective assessment in residency-based training for transoral robotic surgery," *The Laryngoscope*, vol. 122, no. 10, pp. 2184–2192, 2012.

[10] S. S. Vedula, A. Malpani, N. Ahmidi, S. Khudanpur, G. Hager, and C. C. G. Chen, "Task-level vs. segment-level quantitative metrics for surgical skill assessment," *Journal of surgical education*, vol. 73, no. 3, pp. 482–489, 2016.

[11] Z. Chen, A. Malpani, P. Chalasani, A. Deguet, S. S. Vedula, P. Kazanzides, and R. H. Taylor, "Virtual fixture assistance for needle passing and knot tying," in

2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2016, pp. 2343–2350.

[12] A. Malpani *et al.*, "Automated virtual coach for surgical training," Ph.D. dissertation, Johns Hopkins University, 2017.

[13] L. Tao, E. Elhamifar, S. Khudanpur, G. D. Hager, and R. Vidal, "Sparse hidden markov models for surgical gesture classification and skill evaluation," in *International conference on information processing in computer-assisted interventions*. Springer, 2012, pp. 167–177.

[14] L. Tao, L. Zappella, G. D. Hager, and R. Vidal, "Surgical gesture segmentation and recognition," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI) 2013*, ser. Part III. LNCS, K. Mori, I. Sakuma, Y. Sato, C. Barillot, and N. Navab, Eds. Berlin Heidelberg: Springer, 2013, vol. 8151, pp. 339–346.

[15] S. Sefati, N. J. Cowan, and R. Vidal, "Learning shared, discriminative dictionaries for surgical gesture segmentation and classification," in *Modeling and Monitoring of Computer Assisted Interventions (M2CAI) 2015*. Berlin Heidelberg: Springer, 2015.

[16] C. Lea, G. D. Hager, and R. Vidal, "An improved model for segmentation and recognition of fine-grained activities with application to surgical training

tasks," in *2015 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2015, pp. 1123–1129.

[17] C. Lea, R. Vidal, and G. D. Hager, "Learning convolutional action primitives for fine-grained action recognition," *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[18] C. Lea, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks: A unified approach to action segmentation," in *European Conference on Computer Vision*. Springer, 2016, pp. 47–54.

[19] N. Ahmidi, L. Tao, S. Sefati, Y. Gao, C. Lea, B. Bejar, L. Zappella, S. Khudanpur, R. Vidal, and G. D. Hager, "A dataset and benchmarks for segmentation and recognition of gestures in robotic surgery," *IEEE Transactions on Biomedical Engineering*, 2017.

[20] E. Mavroudi, D. Bhaskara, S. Sefati, H. Ali, and R. Vidal, "End-to-end fine-grained action segmentation and recognition using conditional random field models and discriminative sparse coding," in *Applications of Computer Vision (WACV), 2018 IEEE Winter Conference on*. IEEE, 2018, pp. 1558–1567.

[21] D. Liu and T. Jiang, "Deep reinforcement learning for surgical gesture segmentation and classification," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2018.

BIBLIOGRAPHY

[22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, p. 533, 1986.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[24] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[25] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[26] B. Hammer, "On the approximation capability of recurrent neural networks," *Neurocomputing*, vol. 31, no. 1-4, pp. 107–123, 2000.

[27] A. Graves, "Supervised sequence labelling," in *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, 2012.

[28] H. Siegelmann and E. Sontag, "On the computational power of neural nets," *Journal of Computer and System Sciences*, vol. 50, no. 1, pp. 132–150, 1995.

[29] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016.

[30] S. Hochreiter, "Untersuchungen zu dynamischen neuronalen netzen," *Diploma, Technische Universität München*, p. 91, 1991.

BIBLIOGRAPHY

[31] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," *International Conference on Machine Learning (ICML)*, vol. 28, pp. 1310–1318, 2013.

[32] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[33] K. Cho, B. van Merriënboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," *EMNLP*, 2014.

[34] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, 2016.

[35] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[36] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," UPenn, Tech. Rep., 2001.

[37] C. Sutton and A. McCallum, *An introduction to conditional random fields for relational learning*. MIT Press, 2006, vol. 2.

BIBLIOGRAPHY

[38] C. Lea, R. Vidal, and G. D. Hager, "Learning convolutional action primitives from multimodal timeseries data," in *Proceedings of the IEEE international conference on robotics and automation—ICRA*, 2016.

[39] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[40] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[41] Y. Gao, S. S. Vedula, C. E. Reiley, N. Ahmidi, B. Varadarajan, H. C. Lin, L. Tao, L. Zappella, B. Bejar, D. D. Yuh, C. C. G. Chen, R. Vidal, S. Khudanpur, and G. D. Hager, "Language of surgery: A surgical gesture dataset for human motion modeling," in *Modeling and Monitoring of Computer Assisted Interventions (M2CAI) 2014*. Boston, USA: Springer, 2014.

[42] Y. Gao, S. S. Vedula, G. I. Lee, M. R. Lee, S. Khudanpur, and G. D. Hager, "Unsupervised surgical data alignment with application to automatic activity annotation," *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, 2016.

[43] F. Hutter, H. Hoos, and K. Leyton-Brown, "An efficient approach for assessing hyperparameter importance," in *International Conference on Machine Learning*, 2014, pp. 754–762.

BIBLIOGRAPHY

[44] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[45] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[46] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[47] M. Arjovsky, S. Amar, and Y. Bengio, "Unitary evolution recurrent neural networks," *International Conference on Machine Learning (ICML)*, 2016.

[48] T. Lin, B. G. Horne, P. Tino, and C. L. Giles, "Learning long-term dependencies in NARX recurrent neural networks," *IEEE Transactions on Neural Networks*, vol. 7, no. 6, pp. 1329–1338, 1996.

[49] R. Soltani and H. Jiang, "Higher order recurrent neural networks," *arXiv preprint arXiv:1605.00064*, 2016.

[50] S. Zhang, Y. Wu, T. Che, Z. Lin, R. Memisevic, R. Salakhutdinov, and Y. Bengio, "Architectural complexity measures of recurrent neural networks," *Advances in neural information processing systems (NIPS)*, 2016.

BIBLIOGRAPHY

[51] J. Koutnik, K. Greff, F. Gomez, and J. Schmidhuber, "A clockwork RNN," *International Conference on Machine Learning (ICML)*, pp. 1863–1871, 2014.

[52] J. Schmidhuber, "Learning complex, extended sequences using the principle of history compression," *Neural Computation*, vol. 4, no. 2, pp. 234–242, 1992.

[53] S. El Hihi and Y. Bengio, "Hierarchical recurrent neural networks for long-term dependencies." *Advances in neural information processing systems (NIPS)*, 1995.

[54] J. Martens and I. Sutskever, "Learning recurrent neural networks with hessian-free optimization," in *International Conference on Machine Learning (ICML)*, 2011.

[55] T. A. Plate, "Holographic recurrent networks," *Advances in neural information processing systems (NIPS)*, 1993.

[56] I. Danihelka, G. Wayne, B. Uria, N. Kalchbrenner, and A. Graves, "Associative long short-term memory," *International Conference on Machine Learning (ICML)*, 2016.

[57] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[58] J. Weston, S. Chopra, and A. Bordes, "Memory networks," *International Conference on Learning Representations (ICLR)*, 2015.

BIBLIOGRAPHY

[59] M. Henaff, A. Szlam, and Y. LeCun, "Orthogonal RNNs and long-memory tasks," *International Conference on Machine Learning (ICML)*, 2016.

[60] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[61] ——, "Maximizing long-term gas industry profits in two minutes in lotus using neural network methods," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 2, pp. 315–333, 1989.

[62] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *ICLR*, 2015.

[63] Q. V. Le, N. Jaitly, and G. E. Hinton, "A simple way to initialize recurrent networks of rectified linear units," *arXiv preprint arXiv:1504.00941*, 2015.

[64] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," *International Conference on Machine Learning (ICML)*, 2015.

[65] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[66] T. Cooijmans, N. Ballas, C. Laurent, and A. Courville, "Recurrent batch normalization," *arXiv preprint arXiv:1603.09025*, 2016.

[67] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM. NIST speech disc 1-1.1," *NASA STI/Recon technical report*, 1993.

[68] K.-F. Lee and H.-W. Hon, "Speaker-independent phone recognition using hidden Markov models," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 11, pp. 1641–1648, 1989.

[69] A. K. Halberstadt, "Heterogeneous acoustic measurements and multiple classifiers for speech recognition," Ph.D. dissertation, Massachusetts Institute of Technology, 1998.

[70] C. Chatzaki, M. Pediaditis, G. Vavoulas, and M. Tsiknakis, "Human daily activity and fall recognition using a smartphone's acceleration sensor," *International Conference on Information and Communication Technologies for Ageing Well and e-Health*, 2016.

[71] D. Mumford, "On the computational architecture of the neocortex," *Biological cybernetics*, vol. 66, no. 3, pp. 241–251, 1992.

[72] R. P. Rao and D. H. Ballard, "Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects," *Nature neuroscience*, vol. 2, no. 1, p. 79, 1999.

BIBLIOGRAPHY

[73] K. Friston, "A theory of cortical responses," *Philosophical transactions of the Royal Society B: Biological sciences*, vol. 360, no. 1456, pp. 815–836, 2005.

[74] Y. Huang and R. P. Rao, "Predictive coding," *Wiley Interdisciplinary Reviews: Cognitive Science*, vol. 2, no. 5, pp. 580–593, 2011.

[75] A. Clark, "Perceiving as predicting," *Perception and its modalities*, pp. 23–43, 2014.

[76] Y. Gao, S. S. Vedula, G. I. Lee, M. R. Lee, S. Khudanpur, and G. D. Hager, "Query-by-example surgical activity detection," *International journal of computer assisted radiology and surgery*, vol. 11, no. 6, pp. 987–996, 2016.

[77] F. Despinoy, D. Bouget, G. Forestier, C. Penet, N. Zemiti, P. Poignet, and P. Jannin, "Unsupervised trajectory segmentation for surgical gesture recognition in robotic training," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 6, pp. 1280–1291, 2016.

[78] S. Krishnan, A. Garg, S. Patil, C. Lea, G. Hager, P. Abbeel, and K. Goldberg, "Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning," *International Journal of Robotics Research*, vol. 36, no. 13-14, 2017.

[79] A. Zia, C. Zhang, X. Xiong, and A. M. Jarc, "Temporal clustering of surgical

activities in robot-assisted surgery," *International journal of computer assisted radiology and surgery*, vol. 12, no. 7, pp. 1171–1178, 2017.

[80] S. Bodenstedt, M. Wagner, D. Katić, P. Mietkowski, B. Mayer, H. Kenngott, B. Müller-Stich, R. Dillmann, and S. Speidel, "Unsupervised temporal context learning using convolutional neural networks for laparoscopic workflow analysis," *arXiv preprint arXiv:1702.03684*, 2017.

[81] G. Yengera, D. Mutter, J. Marescaux, and N. Padoy, "Less is more: surgical phase recognition with less annotations through self-supervised pre-training of cnn-lstm networks," *arXiv preprint arXiv:1805.08569*, 2018.

[82] T. Yu, D. Mutter, J. Marescaux, and N. Padoy, "Learning from a tiny dataset of manual annotations: a teacher/student approach for surgical phase recognition," *arXiv preprint arXiv:1812.00033*, 2018.

[83] C. M. Bishop, "Mixture density networks," Aston University, Tech. Rep., 1994.

[84] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[85] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[86] R. DiPietro, N. Ahmidi, A. Malpani, M. Waldram, G. I. Lee, M. R. Lee, S. S. Vedula, and G. D. Hager, "Segmenting and classifying activities in robot-

assisted surgery with recurrent neural networks," *International journal of computer assisted radiology and surgery*, 2019.

[87] A. J. Van Der Schaft and J. M. Schumacher, *An introduction to hybrid dynamical systems*. Springer London, 2000.

[88] A. H. Mirza and S. Cosan, "Computer network intrusion detection using sequential lstm neural networks autoencoders," in *2018 26th Signal Processing and Communications Applications Conference (SIU)*, 2018.

[89] Y. Jia, C. Zhou, and M. Motani, "Spatio-temporal autoencoder for feature learning in patient data with missing observations," in *IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2017.

[90] Y. S. Chong and Y. H. Tay, "Abnormal event detection in videos using spatiotemporal autoencoder," in *International Symposium on Neural Networks*, 2017.

[91] Z. Zhang, F. Ringeval, J. Han, J. Deng, E. Marchi, and B. Schuller, "Facing realism in spontaneous emotion recognition from speech: Feature enhancement by autoencoder with LSTM neural networks," *Interspeech 2016*, 2016.

[92] E. Marchi, F. Vesperini, F. Eyben, S. Squartini, and B. Schuller, "A novel approach for automatic acoustic novelty detection using a denoising autoencoder

with bidirectional lstm neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[93] A. Gensler, J. Henze, B. Sick, and N. Raabe, "Deep learning for solar power forecasting–an approach using autoencoder and LSTM neural networks," in *IEEE international conference on systems, man, and cybernetics (SMC)*, 2016.

[94] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length and helmholtz free energy," in *Advances in neural information processing systems*, 1994, pp. 3–10.

[95] A. Rahman, D. Smith, J. Hills, G. Bishop-Hurley, D. Henry, and R. Rawnsley, "A comparison of autoencoder and statistical features for cattle behaviour classification," in *2016 international joint conference on neural networks (IJCNN)*. IEEE, 2016, pp. 2954–2960.

[96] S. Amiriparian, M. Freitag, N. Cummins, and B. Schuller, "Sequence to sequence autoencoders for unsupervised representation learning from audio," in *Proc. of the DCASE 2017 Workshop*, 2017.

[97] R. DiPietro and G. D. Hager, "Unsupervised learning for surgical motion by learning to predict the future," in *International conference on medical image computing and computer-assisted intervention*. Springer, 2018, pp. 281–288.

[98] ——, "Automated surgical activity recognition with one labeled sequence," in

International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, 2019, pp. 458–466.

[99] ——, "Deep learning: RNNs and LSTM," in *Handbook of Medical Image Computing and Computer Assisted Intervention*. Elsevier, 2020, pp. 503–519.

[100] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," *arXiv preprint arXiv:1206.6392*, 2012.

[101] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, 2015, pp. 802–810.

[102] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[103] A. Radford, R. Jozefowicz, and I. Sutskever, "Learning to generate reviews and discovering sentiment," *arXiv preprint arXiv:1704.01444*, 2017.

[104] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365*, 2018.

BIBLIOGRAPHY

[105] J. Butepage, M. J. Black, D. Kragic, and H. Kjellstrom, "Deep representation learning for human motion prediction and classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6158–6166.

[106] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[107] F. Locatello, S. Bauer, M. Lucic, S. Gelly, B. Schölkopf, and O. Bachem, "Challenging common assumptions in the unsupervised learning of disentangled representations," *arXiv preprint arXiv:1811.12359*, 2018.

[108] M. I. Garrido, J. M. Kilner, K. E. Stephan, and K. J. Friston, "The mismatch negativity: a review of underlying mechanisms," *Clinical neurophysiology*, vol. 120, no. 3, pp. 453–463, 2009.

[109] T. Baldeweg, "Erp repetition effects and mismatch negativity generation: a predictive coding perspective," *Journal of Psychophysiology*, vol. 21, no. 3-4, pp. 204–213, 2007.

[110] M. Mignard and J. G. Malpeli, "Paths of information flow through visual cortex," *Science*, vol. 251, no. 4998, pp. 1249–1251, 1991.

[111] P. Murphy and A. Sillito, "Corticofugal feedback influences the generation of length tuning in the visual pathway," *Nature*, vol. 329, no. 6141, p. 727, 1987.

[112] J. H. Sandell and P. H. Schiller, "Effect of cooling area 18 on striate cortex cells in the squirrel monkey." *Journal of Neurophysiology*, vol. 48, no. 1, pp. 38–48, 1982.

[113] A. Alink, C. M. Schwiedrzik, A. Kohler, W. Singer, and L. Muckli, "Stimulus predictability reduces responses in primary visual cortex," *Journal of Neuroscience*, vol. 30, no. 8, pp. 2960–2966, 2010.

[114] L. Melloni, C. M. Schwiedrzik, N. Müller, E. Rodriguez, and W. Singer, "Expectations change the signatures and timing of electrophysiological correlates of perceptual awareness," *Journal of Neuroscience*, vol. 31, no. 4, pp. 1386–1396, 2011.

[115] J. Ross, D. R. Badcock, and A. Hayes, "Coherent global motion in the absence of coherent velocity signals," *Current Biology*, vol. 10, no. 11, pp. 679–682, 2000.

[116] Z. Kourtzi and N. Kanwisher, "Activation in human mt/mst by static images with implied motion," *Journal of cognitive neuroscience*, vol. 12, no. 1, pp. 48–55, 2000.

[117] B. Krekelberg, S. Dannenberg, K.-P. Hoffmann, F. Bremmer, and J. Ross, "Neural correlates of implied motion," *Nature*, vol. 424, no. 6949, p. 674, 2003.

[118] M. Ranzato, A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra, "Video (language) modeling: a baseline for generative models of natural videos," *arXiv preprint arXiv:1412.6604*, 2014.

[119] V. Michalski, R. Memisevic, and K. Konda, "Modeling deep temporal dependencies with recurrent grammar cells""," in *Advances in neural information processing systems*, 2014, pp. 1925–1933.

[120] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh, "Action-conditional video prediction using deep networks in atari games," in *Advances in neural information processing systems*, 2015, pp. 2863–2871.

[121] N. Kalchbrenner, A. van den Oord, K. Simonyan, I. Danihelka, O. Vinyals, A. Graves, and K. Kavukcuoglu, "Video pixel networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1771–1779.

[122] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.

[123] C. Finn, I. Goodfellow, and S. Levine, "Unsupervised learning for physical interaction through video prediction," in *Advances in neural information processing systems*, 2016, pp. 64–72.

[124] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," *arXiv preprint arXiv:1511.05440*, 2015.

[125] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, 2019.

[126] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.

[127] R. B. Palm, "Prediction as a candidate for learning deep hierarchical models of data," *Technical University of Denmark*, vol. 5, 2012.

[128] R. Chalasani and J. C. Principe, "Deep predictive coding networks," *arXiv preprint arXiv:1301.3541*, 2013.

[129] W. Lotter, G. Kreiman, and D. Cox, "Deep predictive coding networks for video prediction and unsupervised learning," *arXiv preprint arXiv:1605.08104*, 2016.

[130] M. Längkvist, L. Karlsson, and A. Loutfi, "A review of unsupervised feature learning and deep learning for time-series modeling," *Pattern Recognition Letters*, vol. 42, pp. 11–24, 2014.

[131] P. Malhotra, V. TV, L. Vig, P. Agarwal, and G. Shroff, "Timenet: Pre-trained deep recurrent neural network for time series classification," *arXiv preprint arXiv:1706.08838*, 2017.

[132] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using LSTMs," in *International conference on machine learning*, 2015, pp. 843–852.

[133] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[134] R. DiPietro, C. Lea, A. Malpani, N. Ahmidi, S. S. Vedula, G. I. Lee, M. R. Lee, and G. D. Hager, "Recognizing surgical activities with recurrent neural networks," *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 551–558, 2016.

[135] R. DiPietro, C. Rupprecht, N. Navab, and G. D. Hager, "Analyzing and exploiting NARX recurrent neural networks for long-term dependencies," *ICLR Workshop Track*, 2017.

[136] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 1019–1027.

[137] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[138] Y. Mao and Z. Yin, "A hierarchical convolutional neural network for mitosis

detection in phase-contrast microscopy images," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2016, pp. 685–692.

[139] Y. Miao, M. Gowayyed, and F. Metze, "EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding," *Automatic Speech Recognition and Understanding (ASRU)*, pp. 167–174, 2015.

[140] A. V. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," *International Conference on Machine Learning (ICML)*, 2016.

[141] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[142] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[143] I. Sutskever, *Training recurrent neural networks*. University of Toronto Toronto, Ontario, Canada, 2013.

[144] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," in *Readings in speech recognition*. Elsevier, 1990, pp. 393–404.

[145] P. J. Werbos, "Applications of advances in nonlinear sensitivity analysis," in *System modeling and optimization*. Springer, 1982, pp. 762–770.

[146] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[147] Y. Xie, Z. Zhang, M. Sapkota, and L. Yang, "Spatial clockwork recurrent neural network for muscle perimysium segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2016, pp. 185–193.

[148] C. Lea, R. Vidal, and G. D. Hager, "Learning convolutional action primitives for fine-grained action recognition," *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[149] F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count," *Neural Networks, IJCNN*, vol. 3, 2000.

[150] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org.

BIBLIOGRAPHY

[151] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," *Neural networks: Tricks of the trade*, 2012.

[152] P. J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," *Neural networks*, vol. 1, no. 4, pp. 339–356, 1988.

[153] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.

[154] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[155] S. Sukhbaatar, J. Weston, R. Fergus *et al.*, "End-to-end memory networks," *Advances in neural information processing systems (NIPS)*, 2015.

[156] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Gated feedback recurrent neural networks," *International Conference on Machine Learning (ICML)*, 2015.

[157] J. Chung, S. Ahn, and Y. Bengio, "Hierarchical multiscale recurrent neural networks," *arXiv preprint arXiv:1609.01704*, 2016.

[158] D. Krueger, T. Maharaj, J. Kramár, M. Pezeshki, N. Ballas, N. R. Ke, A. Goyal, Y. Bengio, H. Larochelle, A. Courville *et al.*, "Zoneout: Regularizing rnns by randomly preserving hidden activations," *arXiv preprint arXiv:1606.01305*, 2016.

BIBLIOGRAPHY

[159] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[160] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE transactions on acoustics, speech, and signal processing*, vol. 37, no. 3, pp. 328–339, 1989.

[161] Y. Wu, S. Zhang, Y. Zhang, Y. Bengio, and R. R. Salakhutdinov, "On multiplicative integration with recurrent neural networks," *Advances in neural information processing systems (NIPS)*, 2016.

[162] B. B. Haro, L. Zappella, and R. Vidal, "Surgical gesture classification from video data," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2012, pp. 34–41.

[163] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006.

[164] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[165] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[166] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[167] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[168] M. Malinowski, M. Rohrbach, and M. Fritz, "Ask your neurons: A neural-based approach to answering questions about images," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1–9.

[169] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014.

[170] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling." in *INTERSPEECH*, 2014, pp. 338–342.

Vita

Robert DiPietro received his BS in applied physics and his MS in electrical engineering from Northeastern University; and he continued his MS work for 3 years as an associate research staff member at MIT Lincoln Laboratory. Next, he joined the PhD program at Johns Hopkins, where he has been advised by Gregory D. Hager. Robert's current research focuses on unsupervised representation learning and data-efficient segmentation and classification for time-series data, primarily within the domain of robot-assisted surgery. Robert has published in journals and venues as diverse as the Journal of Applied Physics, Optical Engineering, IEEE Signal Processing Magazine, the International Conference on Computer Vision (ICCV), and the International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI). During his PhD, he received a Louis M. Brown Fellowship, an Intuitive Surgical Fellowship, a Link Foundation Fellowship, and an Excellence in Teaching Award. He is excited to be joining NVIDIA in January of 2020.